UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Contact Recommendation: Effects on the Evolution of Social Networks

Doble Máster Universitario en Ingeniería Informática y en Investigación e Innovación en Tecnologías de la Información y las Comunicaciones

Autor: SANZ-CRUZADO PUIG, Javier

Tutor: CASTELLS AZPILICUETA, Pablo

FECHA: Febrero, 2017

Resumen

En los últimas dos décadas y media, el desarrollo y crecimiento de los sistemas de recomendación ha progresado cada vez más rápido. Esta expansión ha dado lugar a la confluencia entre las tecnologías de recomendación y otras áreas adyacentes, y, en particular, con las tecnologías de redes sociales, que han experimentado un crecimiento exponencial en los últimos años. El presente trabajo explora uno de los problemas más novedosos que surgen de la confluencia entre ambas áreas: la recomendación de contactos en redes sociales.

Nuestro trabajo se centra, por un lado, en obtener una perspectiva completa de la efectividad de una amplia selección de algoritmos de recomendación, incluyendo algunas contribuciones originales, y considerando perspectivas novedosas que van más allá del acierto de la recomendación. Por otro, en el estudio de la influencia que los algoritmos de recomendación de contactos ejercen sobre la evolución de las redes sociales y sus propiedades. Una fracción no despreciable de los nuevos enlaces que aparecen en las modernas redes sociales online (como Twitter, LinkedIn o Facebook) son creados a través de sugerencias de contactos personalizadas de la plataforma de red social. Los sistemas de recomendación están convirtiendose en un factor importante para influenciar la evolución de la red. Comprender mejor este efecto y aprovechar la oportunidad de obtener más beneficios de la acción de los recomendadores desde una perspectiva amplia de la red son, por tanto, direcciones de investigación que merece la pena investigar, y que estudiamos aquí.

Nuestro estudio comprende trabajo teórico y algorítmico, incluyendo la definición y adaptación de métricas de evaluación novedosas. Esto lo complementamos con un exhaustivo trabajo experimental, en el que comparamos múltiples algoritmos de recomendación desarrollados en diferentes áreas, incluyendo la predicción de enlaces, los sistemas de recomendación clásicos y la recuperación de información, junto con otros algoritmos propios del campo de recomendación de contactos. Hemos evaluado los efectos en la evolución de las redes sociales mediante experimentos offline sobre varios grafos de la red social Twitter. Hemos considerado dos tipos de grafos: grafos de interacción entre usuarios (retweets, menciones y respuestas) y grafos de amistad explícitos (relaciones de follow). Con dichos experimentos, se ha medido no sólo el acierto de los recomendadores: también se han estudiado perspectivas más novedosas, como la novedad y diversidad de las recomendaciones, y sus efectos sobre las propiedades estructurales de la red.

Finalmente, hemos analizado los efectos de promocionar ciertas métricas globales de diversidad estructural de las recomendaciones sobre el flujo de información que viaja a través de las redes, en términos de la velocidad de la difusión y de la diversidad de la información que reciben los usuarios.

Palabras clave

Recomendación, redes sociales, Twitter, evaluación, novedad, diversidad, diversidad estructural, evolución, difusión, predicción de enlaces.

Abstract

Over the last two and a half decades, the development and expansion of recommender systems has progressed increasingly fast. This expansion has given place to the confluence between recommendation technologies and other adjacent areas, notably social networks technologies, which have similarly experienced an exponential growth of their own in the last few years. This thesis explores one of the most novel problems arised from the confluence between both areas: the recommendation of contacts in social networks.

Our work focuses, on one hand, on gaining a comprehensive perspective of the effectiveness of a wide range of recommendation algorithms including some of our own original contributions, and considering novel target perspectives beyond the recommendation accuracy. And on the other, on the study of the influence that contact recommendation algorithms have on the evolution of social networks and their properties. A non-negligible fraction of the new links between pairs of users in modern online social networks (such as Twitter, Facebook or LinkedIn) are created through personalized contacts suggestions made by the social network platform. Recommender systems are hence becoming an important factor influencing the evolution of the network. Better understanding this efffect, and taking advantage of the opportunity to draw further benefit from the action of recommenders with a broader network perspective, are therefore a worthwile research direction which we aim to undertake here.

Our study comprises algorithmic and theoretical work, including the definition and adaptation of novel evaluation metrics. We complement this with extensive experimental work, where we start by comparing multiple recommendation algorithms developed in different areas including link prediction, classical recommender systems and text information retrieval along with other algorithms from the contact recommendation field. We have evaluated the effects over the evolution of social networks via offline experiments over several graphs extracted from the Twitter social network. Two different types of graphs have been considered: graphs which represent the different interactions between users (retweets, replies and mentions) and explicit graphs (follows relations). With those experiments, we have not only measured the accuracy of the recommendation algorithms, but also more novel perspectives such as the novelty and diversity of the recommendations, and their effects on the structural properties of the network.

Finally, we have measured the effects of enhancing the structural diversity of the recommendation over the flow of information which travels through the network

in terms of the speed of the diffusion and the diversity of the information received by the different users.

Keywords

Recommendation, social networks, Twitter, evaluation, novelty, diversity, structural diversity, evolution, information diffusion, link prediction.

Agradecimientos

En primer lugar, me gustaría agredecer a mi tutor, Pablo Castells, la oportunidad de desarrollar el presente trabajo en el Grupo de Recuperación de Información, así como su constante guía y apoyo para sacar adelante todo esto. También quiero dar las gracias a todos aquellos que han pasado por el grupo en este último año y medio: Rocío, Sofía, Nacho... Gracias a vosotros, este trabajo ha sido más sencillo.

No puedo olvidarme de todos aquellos que me han acompañado en aquellas prácticas del máster que parecían interminables: Dani, Rafa, Carlos, Noemi, Guido,… En especial, de entre todos ellos, me gustaría dar las gracias a Rus por su trabajo, esfuerzo y apoyo ante cualquier problema.

Fuera del ámbito universitario, me gustaría dar las gracias a mis amigos, que me han apoyado durante todos estos años: Nadia, Arabia, Jose… Sin vosotros, esto sería imposible. Finalmente, pero no por ello menos importante, me gustaría dar las gracias a toda mi familia por su apoyo y cariño durante toda mi vida.

Table of Content

Figure Index

Table Index

1. Introduction

1.1 Motivation

The information that can be accessed by the average citizen in the different aspects of his daily life has grown to massive scale. The difficulty of manually handling this information has motivated the growth of personalized recommendation technologies to help in the discovery of products or contents that provide value to the users. Taking into account the individual preferences of each user, recommender systems filter the available information and select those items the user might be interested in, according to their prediction.

Recommender systems have been in development since the early 90s, and their development and expansion have progressed increasingly fast in the last few years. Initially, those systems were mainly oriented to e-commerce, and today, they are pervasive in the most varied areas, beyond the best-known examples such as Amazon (pioneer enterprise in the field), eBay or Walmart. More recently, recommender systems have been integrated in virtually every domain, such as news (Google News), audiovisual content (Netflix, Spotify, Youtube), personalized advertisement (Google AdSense), or software and apps stores (Google Play, Steam).

This expansion has given place to the confluence between recommendation technologies and other adjacent areas, notably social networks technologies, which have similarly experienced an exponential growth in the last few years. This thesis explores one of the most novel problems arised from the confluence between both areas: the recommendation of contacts in social networks. This problem poses an special characteristic in relation to the classical recommendation tasks: in those tasks, items and users were separate objects. However, in this case, the items to recommend are chosen among the set of users, and there is additional information for the recommenders, like the structure of the links and interactions between the users in the network.

We pursue several goals: On one hand, we aim to search, study and analyze the state of the art in the field of contact recommendation in social networks. On the other hand, we explore the definition and implementation of new algorithms, and compare their effectiveness and properties with algorithms previously documented in the literature for recommending contacts. Finally, we provide new perspectives for the evaluation of link recommendation in social networks, related to the novelty and the diversity of the recommendation, as well as the collective benefit.

A specific perspective for the present work consists in the study of the influence that contact recommendation algorithms have in the evolution of social networks and their properties. A great fraction of the new links between pairs of users in social networks like Twitter, LinkedIn or Facebook are created through personalized contact suggestions made by the social network platform, so recommenders systems are hence becoming an important factor influencing the evolution of the network and its properties. Better understanding this efffect, and taking advantage of the opportunity to draw further benefit from the action of recommenders with a broader network perspective, are therefore a worthwile research direction which we aim to undertake

here. The properties of networks can be studied from several perspectives, like the ones which have been in development in the social network analysis fields: density, degree distribution, distances, clustering coefficient, modularity, strength of the links, behavior in propagation phenomena, etc. (Newman et al. 2010, Easley et al. 2010). The connection between the large set of measures provided by social network analysis, and the effect which recommendation may have over them provides a research opportunity which has not been widely explored yet.

1.2 Goals

The main objective of the present work consists in measuring the effects of several contact recommendation algorithms in the evolution of social networks. This main objective is subdivided in the following goals:

- Reproduce and compare previously documented algorithms in the context of user recommendation in social networks, and adapt others which have not been applied in this context yet.
- Differentiate between explicit networks (follows networks in Twitter) and interaction networks (retweet, reply, mention in Twitter). Analyze if the most effective algorithms are the same in both scenarios, or there are differences.
- Explore and analyze the meaning and utility of novelty and diversity metrics in the context of contact recommendations.
- Analyze the effect of the directionality of the edges in the effectiveness of the algorithms. Traditionally, algorithms documented in the literature have focused on undirected graphs. In this thesis, the behavior of the different algorithm variants for directed graphs which may take different directions for the edges will be tested and analyzed.
- Better understand the effects of recommendations in the global evolution of social networks. Use this understanding to apply it to the different algorithms and recommendation strategies so several properties which may be desirable in networks may be optimized. Several novel perspectives which go beyond the accuracy of the recommendations are considered, such as studying novelty and diversity metrics, as well as global properties of the networks, with the goal of improving their characteristics as a whole. We consider social networks as dynamic entities which evolve under different influences. Among them, recommender systems might play an important role.

1.3 Document structure

The present work is divided in 7 chapters and two annexes, which are detailed next:

- **Chapter 1. Introduction:** Motivation and goals of the present work. The notation which will be used in the rest of the document is also described here.
- **Chapter 2. State of the art:** A review of the basic concepts and previous work done in the different areas that this work covers. We focus on two different directions: recommender systems methods and evaluation techniques, focusing on the particular case of social recommendation, and social network analysis techniques.
- **Chapter 3. Recommendation Algorithms:** We thoroughly describe the different recommendation algorithms we use in our research.

- **Chapter 4. Evaluation Metrics:** We introduce the different evaluation perspectives we will use for comparing the recommendation algorithms in our experiments. Also, we will explain in detail the different metrics associated to each perspective.
- **Chapter 5. Experiments**: In this chapter, we exhaustively compare and analyze the effectiveness of several recommendation algorithms in terms of accuracy, novelty, diversity and a novel perspective known as structural diversity, which measures the effects of the recommendation algorithms on the properties of social networks.
- **Chapter 6. Information Diffusion**: In this chapter, we analyze the effects of the structural diversity metrics on the speed and diversity of the information which flows through the network.
- **Chapter 7. Conclusion:** In this chapter, we summarize the contributions of the present document, and propose several research lines to further explore the contact recommendation problem in social networks.
- **Annex I. Derivations:** Mathematical derivations of several new algorithms and metrics.
- **Annex II. Complete Experimental Results:** Complete results of the comparative of contact recommendation algorithms in terms of accuracy, novelty, diversity and structural diversity.

1.4 Notation

$\mathcal{U}$	Set of users of the social network graph.
E	Set of edges of the social network graph.
E_{train}	Edges in the training partition of the graph.
E_{test}	Edges in the test partition of the graph.
A_{ij}	Element in the i-th row and the j-th column of the adjacency matrix of a network.
$\mathcal{Z}$	Set of aspects of the nodes (for diversity metrics).
$\mathcal{C}$	Set of communities of the graph.
$\Gamma(u)$	Set of neighbours of user u. This notation may mean any directionality for the selected edges.
$\Gamma_{in}(u)$	Set of incident nodes to user u. It is the set of users which follow u or have interacted with u.
$\Gamma_{out}(u)$	Set of adjacent nodes to user u. It is the set of users which u is following or u has interacted with.
$\Gamma_{und}(u)$	The union of the sets of incident and adjacent neighbors of user u.
$\|X\|$	Number of elements in the set X.
$f_u(v)$	Recommendation score.
$\mathcal{R}(u)$	Set of recommended contacts to user u.

2. State of the art

The present work studies the contact recommendation problem in social networks. Contact recommendation in social networks convenes the confluence of work in recommender systems and social network analysis. In this chapter, we provide a general overview of the most relevant work in these two fields directly related to the goals of our research.

It should be noted that, in this chapter, we only provide a few details on the algorithms, metrics and techniques used in our research, which will be further detailed in the following chapters of this document.

2.1 Recommender Systems

Recommender systems started to be conceived and developed in the early 90's, and their penetration in everyday applications has been accelerated in recent years. In their begginings, these systems were mainly oriented to e-commerce, and today, recommendation technologies are integrated in most diverse domains including online shopping (Amazon, eBay, Walmart, Fnac, etc.), news (Google News), music and video streaming (Netflix, Spotify, Youtube), personalized advertising (Google AdSense), or app stores (Google Play, Steam). The development of these systems is a multi-disciplinary field, which takes elements of Artificial Intelligence, Human Computer Interaction, Data Mining, Statistics, Marketing or Consumer Behaviour.

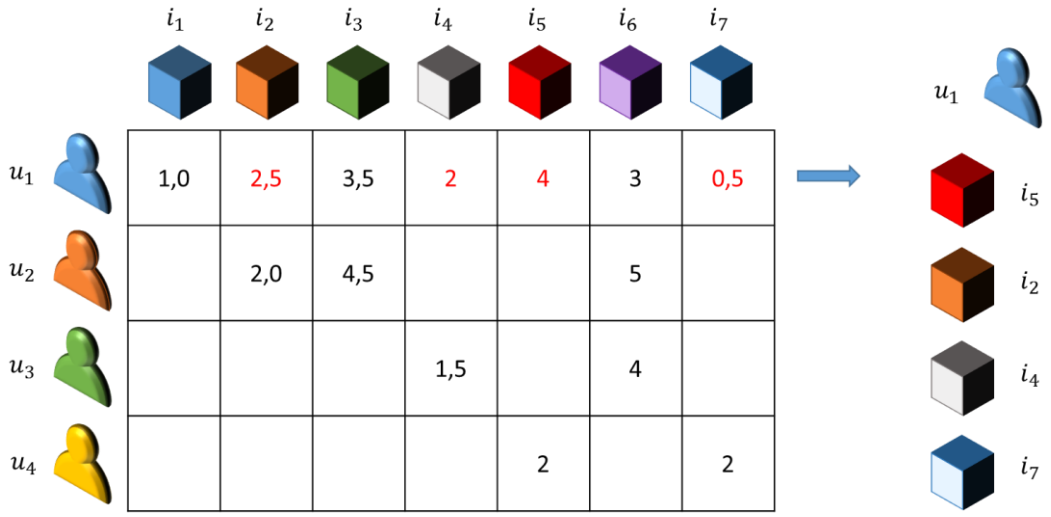


Figure 1. The recommendation task. The scores in red are generated by the recommender system.

Recommender systems are tools which aim to suggest items to users, according to their preferences or necessities. To this end, they seek to predict the utility of the items for the user. In order to do that, the recommendation task is defined as follows: the system can observe a set of users interacting with a set of items. This observation can be recorded in the form of explicit ratings (e.g. the typical 5-star convention as Amazon, Netflix or Google Play or the binary like/dislike feedback as on Facebook, Instagram or

Steam) or implicit (the number of times a user interacts with – plays, clicks, buys, etc. – an item).

User-item feedback can be seen as a user-item matrix like the one in Figure 1, where some cells have observed data (e.g. a rating value), and most do not (the system did not observe any interaction between those user-item pairs). In this view, the recommender's task is to generate a score $f_u(i)$ for each user-item for which no observation was recorded in the matrix. Based on these scores, for each user (called target in this context) the system ranks the different items in descending order. This is exemplified in Figure 1, showing a recommendation for a single user.

2.1.1 Recommendation algorithms

Many recommendation algorithms have been developed since the field took off (Adomavicius et al. 2005, Ricci et al. 2015). Traditionally, recommender systems have been classified in three categories according to the type of input data taken and how it is processed (Adomavicius et al. 2005): content-based methods, collaborative-filtering methods and hybrid methods.

Content-based methods consider that the user is prone to liking items similar to the ones he liked in the past (Adomavicius et al. 2005). These methods analyze the set of rated items by an user and generate a profile for that user in terms of the set of features that describe the items (for example, in a film recommender system, the genres of the films, the director, the actors, etc.). Then, the utility or relevance of an item is computed as a function of the similarity between those users and the profile. Two limitations have been detected for these approaches: first of all, if two items have the same features, they are completely indistinguishable for the recommender system; secondly, the content-based approach may produce an overspecialization of the recommended items: since only similar items to the ones previously consumed are recommended, the recommendation will not encourage the user to discover different elements from the ones already experienced.

Collaborative filtering algorithms (Goldberg et al. 1992) are considered the most popular and widely implemented recommendation strategies (Ricci et al. 2015). These collaborative approach uses the ratings provided by other users to predict the relevance of the items for a certain user. Two different subfamilies of collaborative filtering algorithms are commonly distinguished: neighborhood-based and model-based algorithms (Adomavicius et al. 2005, Koren et al. 2009). Neighborhood-based algorithms (also known as memory-based or heuristic algorithms) generate recommendations according to the ratings given to a certain neighborhood of the user or the item. The neighborhood is a set of users (or items) with similar ratings on some of the same items (or by some of the same users) to the ones of the target user (or the candidate item). Model-based methods build a processed representation (a model) from the raw rating data, and produce recommendations using the model. A particularly successful family of model-based algorithms are the ones based on so-called latent factors, which seek to characterize both users and items on a common latent space inferred from the ratings patterns (Koren et al. 2009). For the items, each factor captures some latent characteristic of the items. For example, in the case of computer games, one latent variable could represent the difficulty of the game for average users. In real applications, a meaning for the latent factors in terms of the items is hardly found, but examples like the previous one work as an intuition of how these approaches work. For users, the elements of the vector measure to what extent the user is interested in items

which have high values in the corresponding factor. A simple example in the context of computer games is shown in Figure 2.

The recommendations generated by collaborative filtering overcome some of the limitations of content-based techniques (Ning et al. 2015). In particular, collaborative filtering tecnhniques do not necessarily recommend similar items to the ones previously consumed by the target user, so the recommender system is less likely to overspecialize. The weak point of collaborative filtering is data sparsity: algorithms take solely as input the ratings given by the users, so new items (with no ratings) cannot be recommended by these methods. Content-based approaches can be a good alternative in such cases.

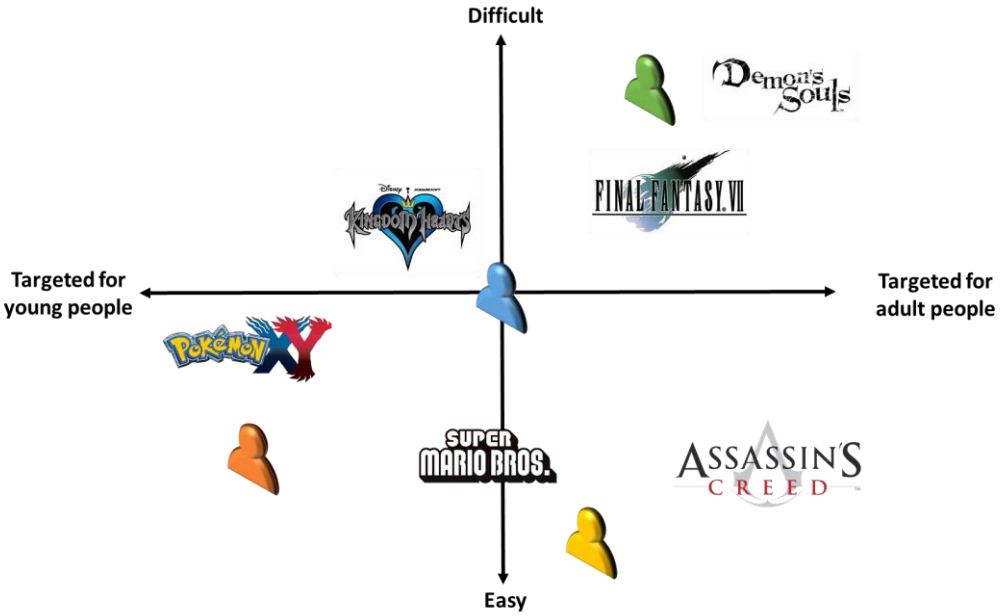


Figure 2. Simplified latent variables example in the context of computer games

Hybrid algorithms, as their name indicates, combine elements from both content-based and collaborative filtering approaches (Adomavicius et al. 2005). These algorithms are created to overcome the disadvantages of both kinds of algorithms. They can be created in many ways, such as combining the outcomes of several recommenders of each one of the types, adding content-based characteristics to a collaborative filtering method, adding collaborative filtering characteristics to a content-based approach, or creating a general unifying model which combines both characteristics.

2.1.2 Evaluation

The development of recommendation algorithms goes hand in hand with their evaluation, to check their quality and utility and compare the properties of different approaches which can be used for the recommendation. Evaluation is peformed by running several tests using the different algorithms we want to compare with real or simulated data. According to the experimental configuration of those tests, we can classify the evaluation experiments in three different types (Shani et al. 2015): offline experiments, user studies and online experiments.

Offline evaluation checks the performance of recommender systems using a pre-collected data set of users choosing or rating items. These experiments assume that the behavior of the collected users will be similar to the one that users in the final system will exhibit (Shani et al. 2015). Since they do not require interactions with users, it is easy to compare the effectiveness of a wide range of algorithms. However, results may

differ from the real ones, since the dataset may be biased, and they do not allow evaluators to obtain feedback about the performance of the system.

This type of evaluation simulates the online process where the system makes recommendations to users, and the user selects or rates the items which they have considered appropriate. To do that, the whole set of ratings in the data set is partitioned into two separate subsets: the training set and the test set. There are many ways to do this partition: randomly selecting ratings, taking all the ratings created before a given date as the training set, etc. An example is illustrated in Figure 3. These sets represent the ratings before and after applying the split. Recommendation algorithms are run over the ratings in the training set. Then, to check the different properties of the system which define its quality, several metrics are computed over the outcome of these algorithms and the ratings in the test set.

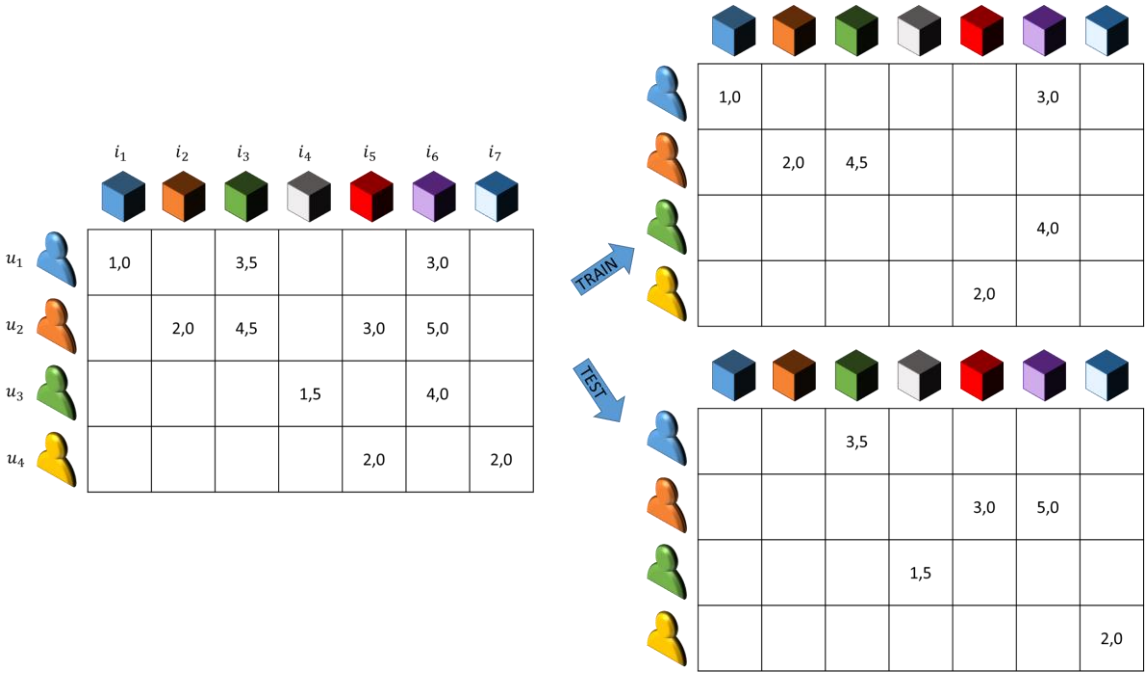


Figure 3. Train/Test data split in offline experiments

User studies specifically recruit a group of users for the purpose of evaluating the system. The users are asked to perform different tasks over the final system, and the actions they make are used to evaluate the recommender system (Shani et al. 2015). This type of evaluation allows observing and recording the behavior of different users when they interact with the system, as well as identifying how recommender systems influence their behavior. Direct feedback from users can be also received by the comments they made while they use the application. However, user studies are expensive to conduct: first of all, a large set of users must be selected to enable significant results, and then, depending on the number and the size of the different tasks, the study can take a long time to complete. Also, the people selection could present some biases which are not present in the set of users of the real system.

Finally, online experiments measure the performance of systems in production, in the real setting. Tipically, these experiments are run to compare several versions of a system: a small fraction of the traffic to the system is randomly redirected to a different recommendation engine, and the interactions with both systems are recorded and compared. Since the evaluation is done in the real system, the results are the most realistic of all the evaluation experiments. However, the experiments are risky, since

irrelevant or bad recommendations provided by the alternative systems may discourage the users from using the real one.

In this work, we will focus only on offline experiments, using different data sets for evaluating different contact recommendation algorithms. As we stated before, in offline experiments, we use several metrics to evaluate different properties of the system. There are many properties which can be studied to determine the quality of a recommender system. The most well-known and developed evaluation perspective is the accuracy one (Shani et al. 2015). This perspective checks how similar are the outcomes of the recommendation algorithms and the real user preferences. We can differ two classes of accuracy measures: ratings metrics and ranking metrics.

Rating metrics

When an evaluator uses ratings metrics to evaluate the the accuracy of the recommender system, he measures how close are the scores given to the different items by the recommender system to the real ones. Several measures have been defined. These ones are useful when the recommendation algorithm produces scores in the same range as the ratings. The most well known (Shani et al. 2015) are the Mean Absolute Error (MAE) of the system:

$$MAE = \frac{1}{|Test|} \sum_{(u,i) \in Test} |f_u(i) - r_u(i)| \tag{2.1}$$

and the Rooted Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{|Test|} \sum_{(u,i) \in Test} \left(f_u(i) - r_u(i)\right)^2} \tag{2.2}$$

Ranking metrics

Ranking metrics take a different perspective, bringing notion of relevance into play: an item is considered relevant for a user if it satisfies a necessity. In the case of recommender system, we consider that an item is relevant if the user likes it. This means the user assigned the item a positive rating, if such explicit feedback is available, or the user simply consumes the item, if only implicit feedback is available. The metrics in this scope are adapted from Information Retrieval (IR), where relevance is a central notion (Baeza-Yates et al. 2010).

Some of these metrics, like Precision or Recall (Baeza-Yates et al. 2010) are simply related to the number of relevant items in the recommendation ranking, while others, like Normalized Cumulative Gain (Järveling et al. 2000) also consider the position of the items in the ranking, giving more importance to relevant items in the top positions of the recommendation ranking.

This kind of metrics are the ones we have used in our work for the evaluation of the accuracy of the contact recommendation algorithms. More detailed information about the different ranking metrics we have used in our research is shown in chapter 4.

Beyond accuracy: novelty and diversity

Providing accurate recommendations is very useful for the user, but this is only one among several important dimensions of recommendation utility. Other technical properties of the system such as scalability, robustness or the privacy of the systems should be considered to make for an overall good user experience (Shani et al. 2015).

But even at a more core and conceptual level, further dimensions matter. Since the beginning of the 2000s, two new properties of the recommendations have been paid increasing attention: novelty and diversity (Castells et al. 2015).

The novelty of a system measures the different between the present and past experiences of the users (Castells et al. 2015). In terms of recommender systems, they measure how different are the items suggested by the system to the ones the user already knows. Two different novelty perspectives have been proposed: the first one, user independent, measures the "anti-popularity" of the recommendations, i.e. how unknown are the recommended items for the different users in the system; the second measures the differences between the recommendation provided to a user and the items the user already knows.

Diversity relates to the differences between the items in recommendation rankings, without considering the past experience of the user. Again, two different perspectives have been studied: first, a local perspective, which measures the distances between the items in each individual recommendation; second, a global perspective, which studies to what extent every item in the system has been recommended.

As far as we know, novelty and diversity have not been applied in the context of contact recommendation. This opens a novel research line which we explore in this work. In chapter 4, we will delve further into both perspectives and their adaptation to the user recommendation task.

2.2 Social Networks

A social network is a set of people or groups of people with some pattern of contacts or interactions between them (Newman 2003). They have been an object of study for different fields like psychology, sociology, biology or statistics. The earliest documented works with explicit notions of social networks were undertaken in the area of social sciences and date back to the last years of the 19th century (Tönnies 1887, Durkheim 1893). The analysis of social networks has many practical uses, such as studying the spread of diseases over a population, understanding how relationships are created, identifying important people in networks, finding latent communities, identifying key connections in the network, predicting social dynamics, or planning marketing campaigns.

The massive transfer of social network information into online platforms starting by the early 2000s opened a whole new horizon for the field, and gave a new meaning to the notion of social network. Platforms like Facebook, Twitter, LinkedIn or Instagram are used every day by hundreds of millions of people worldwide. The availability of visible network data at such an unprecedented scale has multiplied the possibilities for business as much as research, and has boosted the study and exploitation of these networks in the last couple of decades. –what online social networks are to social network science and technology can be compared to what the Web meant for the information retrieval field.

The relationships between different people in social networks can be mathematically modeled as a graph, $G = \langle \mathcal{U}, E \rangle$, where the nodes, $\mathcal{U}$, represent the different individuals, and the edges, E, represent the relationships between users (Easley et al. 2010). These relations can also be seen as a matrix, A, known as the adjacency matrix of the graph, where:

$$A_{ij} = \begin{cases} weight(i,j) > 0 & if \text{ there is a link between nodes } i \text{ and } j \\ 0 & \text{otherwise} \end{cases} \quad (2.3)$$

The weight of the link between two users, i and j, can represent the existence of links between nodes ($weight(i,j) = 1$ for unweighted graphs), or some quantitative property associated to the relationship between i and j, such as how strong the relationship is, the number of times user i has interacted user j, and so forth.

Depending on the nature of the relations between users, we can distinguish two different network types:

- **Directed or asymmetric networks:** These networks represent relationships where interactions between two individuals do not need to be reciprocated. For example, hierarchical relationships inside a company, e-mail networks, or follow networks in Twitter or Instagram. These relations are represented as directed edges in the graph model of the network.
- **Undirected or symmetric networks:** These networks represent relationships like friendship, where the interaction are reciprocal. The interactions are represented as undirected links in the graph model. Online social networks like Facebook or LinkedIn are examples of these networks.

The area of social network analysis is considerably broad and, as noted before, multidisciplinary, and so is the literature. We overview here the work in this area that is most directly relevant for the goals of our research, and to which which we will make reference throughout the present document.

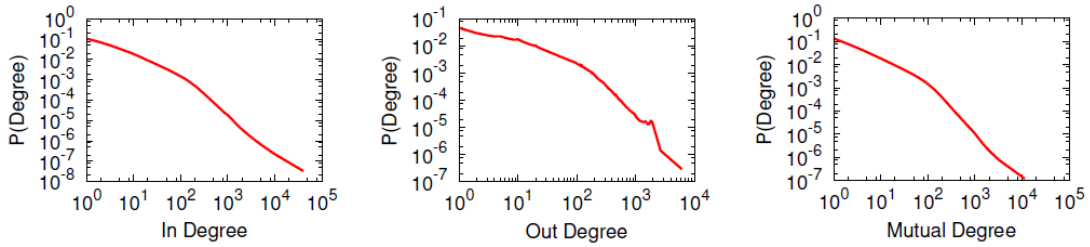
2.2.1 Structural properties of social networks

One of the main tools of social network analysis consists in the study of the structure of the network graphs. Knowing the structure of the graph is useful for finding the most influential users, determining how the network will evolve, etc. The analysis of the structure of real-world networks has led to the observation of several recurring patterns in their structure: small diameter, skewed degree distribution, etc. (Newman 2010). In this section, we explain the main characteristics of those social networks.

Degree distributions

One of the fundamental properties of a network is the distribution of the vertex degrees. The degree of a vertex in the network is the defined as the number of edges that have that vertex as one of their end points. In the case of a directed network, we differentiate the out-degree of the node (the number of outgoing edges) and the in-degree (the number of incoming edges). To study the degree distribution of the networks, it is common to represent the values of the degrees against the proportion of the nodes which have that degree, as shown in Figure 4.

In real-world social networks, it is usual to observe right skewed distributions: most of the nodes have a very low degree, but there is a significant “tail” of the distribution, which corresponds to the nodes with higher degree (Newman 2010) –a few nodes connect to a large fraction of the nodes of the network. Those nodes are known as hubs. In directed social networks, the same stands for the in-degree and out-degree distributions. As an example of this fact, Figure 4, obtained from Myers et al. (2014), shows the in-degree and out-degree distributions of the Twitter graph, as well as the degree distribution of an undirected graph that only contains the links that are reciprocated.



(a) In degree (All) (b) Out degree (All) (c) Mutual degree (All)

Figure 4. Twitter degree distributions in log-log scale (from Myers et al. 2014)

Average shortest path length

The average shortest path length measures the distances between two different nodes in the network. It is highly related to the so-called small-world effect, one of the most widely discussed phenomena in social networks: in a network, the average distance between pairs of nodes (defined as the average length of the shortest paths between the nodes in each pair of nodes) is very small, considering the huge size of the network. This effect was first observed in the Milgram's experiment in the 1960's (Milgram 1967).

Real-world social networks show this phenomenon: for example, the average shortest path length in Twitter in 2012 (a social network with around 175 million users) was around 4.05 steps (Myers et al. 2014), and the Facebook network in 2011 (with 721 million active users) had an average distance of 4.3 steps (Ugander et al. 2011).

Leskovec et al. (2007) also observed that the distance between nodes in networks does not necessarily increase when the network grows. In fact, they observed that many networks reduced their diameter (the maximum distance between two nodes in the graph) as the network grows.

Clustering coefficient

Clustering coefficient is one of the most simple and widely-known graph metrics, and it is related to the transitivity of the network: it measures the proportion of transitive triads in the network. In this context, a triad is defined as a set of three nodes who form a path of length two, i.e. if we name the users, u, v, w, then, they form a triad if u follows v and v follows w. A triad is considered transitive if there is an edge between the starting and ending nodes of the path, i.e. u follows w. Examples of triads are shown in Figure 5.

(a) Non-transitive triad (b) Transitive triad

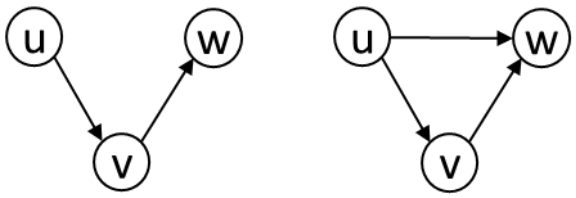


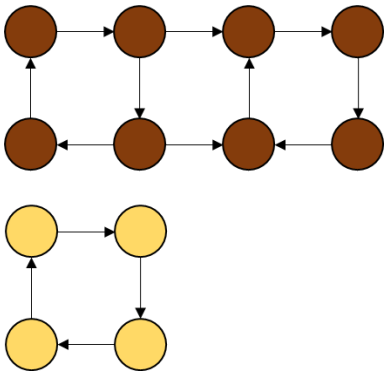
Figure 5. Triads

Connected components

A connected component of a network is a maximal subset of the vertices of the network such that there is, at least, a path from each member of the subset to each other

members. If the network is directed, we differ two types of components: weakly connected components, if we ignore the direction of the edges, and strongly connected components, if we do not. Figure 6 shows an example of a directed graph with two weakly connected components and three strongly connected ones.

a) Weakly connected components



b) Strongly connected components

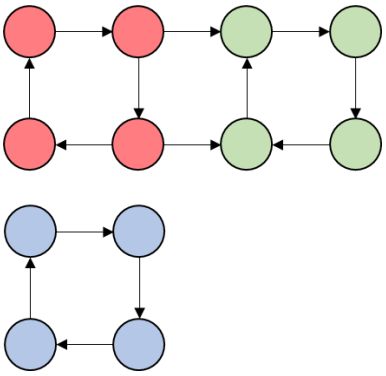


Figure 6. Connected components of a directed graph

Real-world networks usually have a giant component which contains at least half of the nodes of the network, and most often in fact, over 90% (Newman 2010). For example, the giant component in Facebook is estimated to contain 99% of the nodes (Ugander et al. 2011). In directed graphs, this is still true for weakly connected components, but not necessarily for strongly connected ones. As an example, in Twitter, the largest weakly connected component is estimated to contain 92,9% of the users, but the largest strongly connected one only has 68,7% of them (Myers et al. 2014).

2.2.2 Communities

A natural phenomena which occurs in social networks is the spontaneous, explicit or implicit gathering of people in different groups or communities. Communities in a social networks are defined as subsets of nodes with dense connections inside the subset, and sparse connections to people outside that subset (Newman 2006). The formation of these communities is often related to homophily biases (McPherson 2001): contacts between similar people occur at a higher rate than contacts between very different people. The similarities and differences between people may be related to their preferences, location, social position, proffesion, etc.

The detection of communities is one of the most widely studied problems in social network analysis and graph science. Several models and algorithms have been developed for finding and quantifying communities in networks. All these methods, using the structural properties of the graph, seek to find a partition of the network which maximizes the intracommunity interactions and minimizes the intercommunity interactions. As a partition of the network, communities are related to the concept of connected component. However, due to the giant component phenomenon, connected components are highly restrictive respect to the concept of community, which provides plenty of additional information for the analysis of the network. The quality of a partition is usually evaluated by the so-called modularity of the graph (Newman & Girvan 2004). This measure computes the number of links inside communities, in relation to the expected number of links in a random multigraph where the degrees of the nodes are the same as the ones in the original graph. It is defined as

$$\mathrm{mod}(G) = \frac{\sum_{ij}\left(A_{ij} - \frac{|\Gamma(i)||\Gamma(j)|}{m}\right)\delta(c_i, c_j)}{m - \sum_{i,j}\frac{|\Gamma(i)||\Gamma(j)|}{m}\delta(c_i, c_j)} \tag{2.4}$$

where $\Gamma(i)$ represents the set of neighbours of node i, m represents the number of links in the network, c_i represents the community that node i belongs to, $\delta(c_i, c_j) = 1$ when $c_i = c_j$ and $\delta(c_i, c_j) = 0$ otherwise.

It is known that there is always a partition of a graph which achieves maximum modularity. However, finding such optimal partition of the network is computionally unfeasible: it is an NP-Hard problem (Brandes et al. 2008). Many heuristic methods which provide reasonably good results have been developed (Orman et al. 2011). We describe next the main families of such algorithms.

Some algorithms apply a hierarchical divisive approach, based on link centrality measures. These algorithms iteratively remove edges which minimize a certain measure, until separate components of the graph are obtained. Those components are considered the communities of the graph. Several metrics, like the betweenness of the links, i.e. the number of shortest paths in which the link is included (Newman et al. 2004), or the local clustering coefficient of the nodes, i.e. the number of triangles to which the edge belongs (Radicchi et al. 2003) have been used.

Another approach for obtaining an optimal partition consists in greedily optimizing the modularity of the graph. The so-called Louvain algorithm (Blondel et al. 2008) or the FastGreedy algorithm proposed by Clauset et al. (2004) are examples of this approach. Clauset et al. proposed a hierarchical agglomerative algorithm which iteratively joins the pair of communities whose combination produces the largest increase in the modularity value. The Louvain method iteratively increases the modularity by moving users to other communities in the graph and mantaining those changes which produce the largest increase in the modularity of the graph.

Other algorithms take advantage of the matrix formulation of a graph to use linear algebra tools, like eigenvalues and eigenvectors. For example, the Leading Eigenvector approach (Newman 2006) reformulates the modularity optimization problem as an eigenvector finding problem using a so-called modularity matrix.

Another family uses tools derived from information theory to estimate the best partition of the network. Infomap (Rosvall et al. 2008) belongs to this family, and finds an optimal partition by minimizing the quantity of information needed to represent a random walk in the network.

Finally, some algorithms simulate diffusion processes in the network to identify communities. For example, the Label Propagation method (Raghavan et al. 2007) assigns a unique label to each node in the network and, iteratively, each node adopts and propagates the label which a majority of its neighbors has adopted. At the end of the process, nodes with the same labels form the different communities of the graph.

Yang et al. (2016) provide a comparative of eight different community detection algorithms, including some of the previously mentioned ones, in terms of accuracy and compute time. Comparing the outcomes of these algorithms over artificial networks, they found that Infomap and Louvain algorithms provide better communities than the rest of the algorithms, even when the proportion of links between communities is greater than 50%, and with one of the main community detection. Both algorithms work even better when the graph is large. Leading Eigenvector algorithm is quickly

outperformed by the rest of algorithms when the number of edges between communities are detected. In terms of complexity, Infomap, Label Propagation ($\mathcal{O}(E)$) and Louvain ($\mathcal{O}(N\log(N))$ are the fastest approaches, while Girvan-Newman (Newman et al. 2004) is the slowest of all ($\mathcal{O}(NE^2)$). Yang et al. (2016) also empirically show these results.

2.2.3 Strength of links

The strength of a tie between two people is defined as a combination of the amount of time spent on the relation, emotional intensity, intimacy and reciprocal services which characterize a link between those people (Granovetter 1973). Strong links represent e.g. ties with family or close friends, while weak ones may represent ties with people you meet at work, shopkeepers in the local market, etc. The advantages and disadvantages of strong and weak ties have been studied since the beginning of 1970s. One of the most influential and important theories is the one proposed in by Mark Granovetter (1973).

Granovetter hypothesized that contacts maintained via weak ties provide more novel information and resources than the ones maintained through strong ties, playing a major role in the diffusion of information. This is interesting for the analysis of contact recommendation, since recommending weak links to people may have an impact on the novelty and diversity in the flow of information through the network. Granovetter proposed that the novelty of the information and resources comes from a subset of the weak ties called bridges, which provide the only path between two people in the social network. As an additional definition of weak ties, Granovetter also proposed the notion of local bridge: a link in the network which increases the shortest distance between two users in more than one step. This is related to the concept of the redundancy of the links: the number of distance 2 paths between two connected nodes in the network. A local bridge is therefore a link which has no redundancy.

Granovetter's weak tie definitions are too restrictive in practice in real social networks. In fact, Granovetter (1973) stated that both global and local bridges in a network were only a particular definition of the weak links in the network in terms of structural properties, but there could be more of those links in the network. The giant component phenomenon (described in section 2.2.2) makes the connected component decomposition rather irrelevant, and so are therefore the global bridges connecting them. Even the notion of local bridge can be made more informative: Easley et al. (2010) propose to generalize the notion of strength of a link in terms of the neighborhood overlap (or embeddedness) of the link, which is computed as:

$$Embeddedness(u,v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|} \tag{2.5}$$

De Meo et al. (2014) proposed a further extension to the concept of global weak tie in terms of its structural properties: he defined as a weak link every edge between two different communities in the graph. Since communities are always restricted to a single connected component, every link between two different components (global bridge) is still considered a weak link in this definition, posing a natural extension to that concept.

2.2.4 Evolution

Social networks are highly dynamic objects, which change over time with the arrival of new people and the development of new interactions between existing nodes in the network. Discovering and understanding the mechanisms in the evolution of those networks over time is one of the prominent problems addressed by network science (Liben-Nowell et al. 2003).

The studies of the evolution of social networks roughly follow two main approaches: a) the creation of mathematical models that describe the formation and evolution of the network; and b) predicting which edges will form next among the users in the network. Since the second approach, known as link prediction, has also more interpretations, we will describe it in a separate section. In this section, we briefly recall some of the most interesting evolutionary models in social networks.

As we stated, a common approach for studying the evolution of social networks consists in the formulation of simplified mathematical models that describe the formation of macroscopic structural graph properties arising through the addition of nodes and links, such as skewed degree distributions, high clustering coefficients, or small diameters (Newman 2003). Modelling graphs has many uses (Kumar et al. 2000): many problems may be computationally dificult for general and real graphs, but with a suitable model, we can design, analyze and simulate algorithms under that model instead of trying them over the real networks. The goal is thus for graph models to capture some relevant aspect of the real networks. Furthermore, models may suggest unexpected properties of real graphs which can be verified and exploited.

The simplest model is the so-called random model, proposed by Erdös & Rényi (1959). In this model, starting from a fixed number of nodes, links are created randomly between the different pairs of users. Although it is simple, this model is very limited: it does not allow the addition of new nodes, and the degree distribution of the graph follows a Poisson distribution (which differs from the skewed distributions of social networks).

One of the first, most influential and well-known models is the Preferential Attachment model proposed by Barabàsi & Albert (1999), which provides a simple explanation for the formation of skewed degree distributions. In this model, new nodes progressively join the network, creating links to other nodes with proportional probability to the degree of those nodes. Another method for explaining the skewed degree distribution of real-world networks is the vertex copying model proposed by Kleinberg et al. (1999A). This model states that, if a new user in the network follows someone in the network, it is likely to follow at least a subset of the nodes the followee follows. For each new user, this model selects a node at random, and copies a subset of its outgoing links.

Leskovec et al. (2007) proposed a model for studying two empirical phenomena which occur in the evolution real-world networks: the densification of the degree of the graph (networks increase their average degree, following a power-law pattern as they grow), and the reduction of the effective diameter of the network. They proposed the forest fire model, which exhibits both features. The idea behind this model is the following: a new node u creates a link to an existing node in the network, v. The latter may know users which are of interest to u, so user u explores the set of followees of node v, randomly linking to a subset of them. Some of the followers of those new followees may in turn be interesting for the user, so a subset of them is selected as additional followees, and so forth. This process is repeated recursively until no new nodes are discovered.

Finally, another interesting evolutionary model is the one proposed by Leskovec et al. (2008), where a set of new users is introduced in the network according to an “arrival function”, which determines the number of those users. Every new user creates a link to an existing node selected with a probability proportional to the degree of the users (as in the preferential attachment model). Once a link is created, the node which has created it

sleeps for a certain amount of time (different each time a link is created) exponentially distributed. After that time elapses, existing nodes may create a link to a random node at distance 2, if there is one (this increases the local clustering coefficient of the network). Existing nodes may stop creating links at some point. The time interval after their creation when each node stops creating new links is also exponentially distributed, but using a different distribution than the one used for the selection of the sleeping time.

This model is interesting for two reasons: first, instead of focusing on global properties of the graphs, the model focuses on the local properties, such as the local clustering coefficient (Watts et al. 1998); second, the model does not assume that users are always active on networks: the model considers that, similarly to real networks, a user may be active only during certain hours of the day, and that person may even stop creating new links to other users some time after entering the network.

2.2.5 Link prediction

The link prediction problem (Liben-Nowell et al. 2003) studies how new links are created in a network. The most common formulation of this problem is the following: given a snapshot of a network in time t, we want to find a set of links $E_{pred} \subseteq (\mathcal{U} \times \mathcal{U}) \setminus E$ which will be created from t to a future time, t', using features obtained from the network itself.

Link prediction in networks has many uses: discovering missing (existing but non-observed) links in networks (for example, discover crime networks when there is no explicit evidence of it), identifying spurious links in the network or, most relevant for our work, recommending contacts in social networks. Two main different approaches have been followed for addressing the link prediction problem: supervised methods and unsupervised methods.

Unsupervised methods are the most widely used approach to this problem. These methods generate scores for each of the possible new links in the network, and order them in descending order. Many posibilities have been documented for generating those scores. Liben-Nowell et al. (2003) and Lü et al. (2009) propose many methods for link prediction based on different topological properties of the graph: methods based on the common neighbors of the endpoints of the graph, like Jaccard or Adamic-Adar, methods based on the paths which go through the endpoints of the graph, like Katz or matrix forest index; or methods based on random walks like hitting time, rooted PageRank or SimRank. Several probabilistic models, which compute the scores based on the probability of existence of each one of those nodes have also been proposed (Lü et al. 2009), such as the probabilistic relational model, the probabilistic entity relationship model or the stochastic relational model. Finally, methods based on latent factors have also been used for computing the scores (Menon et al. 2011).

Supervised methods consider the link prediction problem as a classification problem on the set of all pairs of nodes, with two classes: presence or absence of a link. Approaches have been proposed that use machine learning classifiers, such as decision trees, support vector machines, multilayer perceptrons or random forests (Al Hasan et al. 2006, Lichtenwalter et al. 2010). For training the classifier, a fraction of the graph links is selected. Then, a set of patterns for every pair of nodes in the network is created, using characteristics of each node, scores of unsupervised methods, etc. Those patterns are used for training. Then, the same patterns are computed for all non-adjacent nodes in the graph for predicting the new links.

These methods have a major problem to address: the imbalance of the classes (Lichtenwalter et al. 2010). Real networks are very sparse, so the majority of the user pairs belong to the "no link" class. Several methods have been proposed for overcoming this problem. For example, Al Hasan et al. (2006) propose using machine learning techniques adapted to imbalanced problems, and Lichtenwalter et al. (2010) propose the use of the SMOTE algorithm (Chawla et al. 2002) for generating new patterns for the minority class.

The link prediction problem has also been generalized to use information about how the network has evolved over time. For example, Gao et al. (2011) propose a latent factor model which considers the state of the network in different points of time.

Since unsupervised link prediction methods are more widely known, and they are closer to the contact recommendation task, we have selected the most representative of them for our research. Further details on this selection are given in chapter 3.

2.2.6 Diffusion

Virtually all online social networks provide means for individuals to exchange information. For example, Twitter allows publishing posts (tweets) in the users' profiles which can be seen (at least) by all the followers of the publisher, as well as sending private messages directly to another person in the network. Users which have received this information may share it using similar mechanisms, making it available to further people, which might in turn share it with more and more users, and so forth, thus giving rise to flows and cascades of information spread through the network.

Information diffusion is defined as the process by which a piece of information (knowledge) is spread and reaches individuals through interactions (Zafarani et al. 2014). This process involves three different elements:

- **Sender**: The source person or group people who start the process by entering (posting, communicating, etc.) the information into the system in the first place.
- **Receiver:** The person or group of people who receive the diffused information.
- **Medium:** The medium through which the diffusion takes place. In the case of information diffusion in social networks, the medium can be the links between individuals in the network.

Information diffusion has attracted the attention of many fields, such as sociology, computer science, biology, economics, or physics. For example, sociology studies how innovations are spread through a population (Rogers 1975, Zafarani et al. 2014) as well as how users influence each other. Biology and medicine have studied the spread of infectious diseases like HIV or the flu in populations to determine how they can be stopped (Hethcote 2000, Newman 2010, Easley et al. 2010). Marketing companies seek to identify the most influential users for maximizing the spread of a product at the smallest cost (Guille et al. 2013, Kempe et al. 2003).

A simple approach to describe spreading processes through a network is to consider that nodes have two possible states: activated (if the node has received the information and it is willing to propagate it) or not. Then, the process is the successive activation of nodes through the network over time (Guille et al. 2013, Zafarani et al. 2014).

The way information flows through a network depends on two main factors: the diffusion protocol (or diffusion model) and the structure of the network. Diffusion models seek to describe and explain how information flows through networks by

defining an abstract, simplified version of the real processes. Several models have been proposed. Some of the most important ones are the following: Goldenberg et al. (2001) proposed the independent cascade model. In that model, an active node u influences the activation of a neighbor v with a certain probability p_{uv}, which depends only on the sender and the receiver of the information. Kempe et al. (2003) proposed a simpler model, known as the linear threshold model, were a node is only activated if a certain fraction of its neighbors are activated. Finally, Demers et al. (1983) considered three models for the exchange of information between databases which have been used for modelling the spread of rumors through networks (Doerr et al. 2011): the push model, where an active node selects a random set of neighbors and activates them, the pull model, where an inactive node selects a random set of neighbors and gets activated if any of those neighbors are active, and the push-pull model, which combines both approaches.

The effects of the structure of a network in the diffusion dynamics have been also studied by many authors. We may highlight three influent studies in this scope: Goldenberg et al. (2001) showed that the influence of weak ties in the speed of information (defined as the number of active nodes in a certain time) is at least as strong as the influence of strong ties. Doerr et al. (2011) demonstrated that preferential attachment networks (Barabási & Albert 1999) spread rumors using the push-pull model faster than uniform random graphs (Erdös and Rényi 1959) due to the presence of low degree nodes linked to big hubs. Finally, De Meo et al. (2014) observed that removing links between communities reduced the speed of the diffusion in Facebook networks.

2.3 Social Recommendation

Recent years have seen the confluence of recommender systems and social network analysis (link prediction) into new specific recommendation tasks and scenarios where recommendation takes place in the presence of social network data and structures. Social networks provide additional information that recommender systems can take advantage of to produce better recommendations; on the other hand, recommender systems can take on new tasks such as recommending people to each other.

Social recommender systems are defined as a subclass of recommender systems which use social relationships as an additional input (Tang et al. 2013). Several types of items can be recommended in the context of social networks: user-generated contents such as tweets, directly produced in the social media, external items (e.g. products, URLs, etc. mentioned in the media), topics, groups of people… We can classify social recommender systems into two main categories, depending on the nature of the items: contact recommendation, if we recommend other users in the network, and item (or content) recommendation otherwise (Guy et al. 2015). The former problem is naturally the one we focus on in our present work. For the sake of completeness though, we briefly comment on the latter complementary task.

2.3.1 Item recommendation

Social item recommender systems add an additional dimension to the classical recommendation problem: the social graph. The exploitation of social relationships in the recommendation of different items such as films, music, books, news, etc., is still an open area in recommender systems. The most common principle for exploiting network structures in this context is based on the concept of homophily (McPherson 2001): users tend to relate to people with similar preferences. As an example, if we aim to

recommend someone a certain TV series, that person may be more likely to enjoy it (or at least be willing to watch it) if several of his friends are following the series already.

Most of the social recommendation algorithms proposed in the literature are based on classical collaborative filtering approaches (Tang 2013). One of the first works in this area, the recommendation algorithm proposed by Goldbeck (2006) adapted memory-based collaborative filtering, but selecting user neighborhoods based on trust values given by users to their friends in a social network, instead of using rating-based similarity. Another similar experiment was proposed by Konstas et al. (2009), who selects neighborhoods by personalized random walks centered on the target users, improving the results of classical collaborative filtering algorithms.

Other item recommendation approaches are based on latent factors models. An example of this methods is the SoRec recommender system proposed by Ma et al. (2008). This approach adapted the method known as Probabilistic Matrix Factorization (Salakhutdinov et al. 2008) adding a new set of latent factors which represent the social network interactions.

2.3.2 Contact recommendation

Contact recommendation, as has already been stated, represents the main focus of the present investigation. It aims at recommending people which the target user may be interested in befriending (in undirected networks like Facebook or LinkedIn) or following (in directed networks like Twitter or Instagram). This recommendation task is very particular: it is exclusive to social networks, and has the salient and interesting characteristic that the set of users and the set of items fold into each other –that is, they are just the same set.

From a most traditional point of view, contact recommendation may be simply seen as a classical recommendation problem where the set of items is the set of users, and the ratings matrix is the adjacency matrix of the graph. Because of that, classical methods like neighborhood models may be used for solving this problem. However, most part of the algorithms which have been used in this context are only oriented to the recommendation of contacts.

Several methods have been proposed for addressing the problem in this perspective. Methods have been proposed which focus on the topological structure of the network (Golder et al. 2009), random walks (Gupta et al. 2013, Backstrom & Leskovec 2012), user-generated content (Hannon et al. 2010) or matrix factorization (Ma et al. 2008). Next, we will provide an overview of the different methods. We divide them in two different points of view: contact recommendation methods used in online social networks, like Twitter, and research proposals.

Contact recommendation in industry

Until the last few years of the 2000s decade, online sites such as Facebook, LinkedIn or Twitter did not provide any recommender system to simplify the search of new users to follow or befriend. Nowadays, most of them provide contact recommendation functionalities on their sites like LinkedIn ‘People You May Know’ or Twitter ‘Who-To-Follow’ systems. An example of contact recommendation in Twitter can be seen in Figure 7.

Few information is known about the recommendation algorithms used by the most important social media sites. Facebook[1] or LinkedIn[2], for example, have just stated that they use information like common connections, groups or work and education information in their contact recommendation features, but no further details about how they work has been published. Unlike them, Twitter has made public some details about the Who-To-Follow system they use on their site for recommending users: Gupta et al. (2013) offered a vision about the architecture of the system, using the graph processing engine Cassovary, and some information about a system based in the algorithm known as SALSA (Lempel et al. 2001). Later, Goel et al. (2015) showed further details on that algorithm, as well as information about other studied contact recommendation algorithms which use HITS (Kleinberg 1999B) or cosine similarity (Salton 1975), as well as data about the impact and revenue of the implantation of the 'Who-To-Follow' system in their web site and mobile applications.

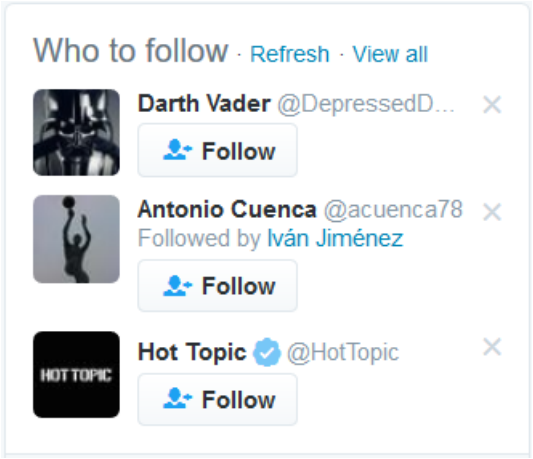


Figure 7. Contact recommendation provided by Twitter Who-To-Follow system

Although the information about these algorithms is not complete, since all these methods have been tested in one of the most important online social networks, and our experiments study the effects of different algorithms over samples of this network, they provide a natural baseline for our experiments. For that reason, we have included all these methods in our comparative. Further details about the adaptation of these algorithms to our work will be provided in section 3.3.

Research proposals

Appart from the industrial approaches, many different research-oriented methods have been proposed. An interesting work in this line is the social recommendation survey written by Ido Guy (2015). Focusing on the contact recommendation problem, Guy states that the problem of recommending people differs depending on several factors like the type of network (directed or undirected) or the network domain (professional networks like LinkedIn or friendship network like Facebook).

One of his most notable contributions to the field is the algorithm known as SONAR. Oriented to friendship recommendation in enterprise social networks, SONAR generates an aggregate score of several characteristics, like the organization hierarchy,

[1] Facebook 'People you may know': https://www.facebook.com/help/501283333222485/ (accessed 02/02/2017

[2] LinkedIn 'People you may know': https://www.linkedin.com/help/linkedin/topics/6096/6118/29/people-you-may-know-feature-overview?lang=en (accessed 02/02/2017)

the number of co-authored papers or patents, mutual friends, comments on enterprise blogs, etc. In their tests, these algorithms improved recommendations made by other algorithms like recommending friends of friends, or content-based recommendation.

Focusing on the recommendations in well-known social networks, like Twitter or Facebook, one of the simplest approaches are the ones proposed by Golder et al. (2009). They proposed four methods for recommending contacts in directed social networks like Twitter based in the concept of homophily: recommending reciprocal links, recommending users which share interests with the target user (i.e. users which follow the same users as the target one), recommending users which share audience with the target user (i.e. users who are followed by the same people than the target user), and recommended people filtered by the preferences of the followees of the target user. Although their proposal is simple, it raises a problem which we will address in the present work: the identification of which neighborhood better describes both the target and the candidate users.

Another interesting approach for recommending contacts in Twitter was the Twittomender system proposed by Hannon et al. (2010). These authors proposed two different approaches to the contact recommendation in Twitter: a content-based approach and a collaborative filtering approach. In the content-based approach, users where represented by their own tweets, the tweets of their followers, the tweets of their followees or a combination of the three set of tweets, using a tf-idf scheme (Baeza-Yates et al. 2011). In the collaborative filtering approach, the tweets in the content based approach were replaced by the identifiers of their followers or their followees (or both).

Finally, several model-based recommender systems have been proposed for the contact recommendation task: Kim et al. (2011) developed the Twittobi system, which applies a probabilistic model to recommend both tweets and users at the same time. Another method was the proposal by Backstrom & Leskovec (2012) for Facebook, which combined supervised learning techniques with random walks. The proposed method's goal is learning a function which receives user profile information (age, place of birth, etc.) and structural information of the network (degree of the nodes, number of frinds, etc.) to generate weights for the different links in the network, so a random walk is more likely to visit nodes the target user is more likely to create links to. Although interesting, these algorithms do not scale well at large sets of users in a single computer.

Online dating

Out of the scope of our research, but related to it, there is an special kind of contact recommender systems which are used in online dating sites like Meetic, Tinder or Lovoo. Although the goal of these systems is also recommending people to people, they have several differences (Pizzato et al. 2010). The first one, and the most important, is the fact that a recommendation is only successful if both target and candidate users like each other. This makes important to consider not only the preferences of the target user, but the preferences of the candidate user at the recommendation. In those user recommendation systems, a social network is not needed, but users have to provide very detailed information to maximize the accuracy of the recommendations. Another difference with respect ot clasical recommendation is that a single user should not be recommended to a lot of users (a single user does not date thousands of people). Finally, after a good recommendation, users may abandon the system forever.

New directions

All the previously mentioned techniques were oriented to maximizing the accuracy of the recommendation. In the last few years, several studies have been done which also consider the effects of the algorithms on the evolution of social networks. Since our work focuses on the study and enhancement of those effects, these studies provide a background for our research. We differentiate two lines:

The first one focuses on the measurement of the effects of recommender systems. The first work in this line was developed by Daly et al. (2010), who empirically tested the effects of four recommendation algorithms on the IBM SocialBlue network, in terms of several properties of the network, like the clustering coefficient or the degree distribution. In 2013, Huang et al. developed a similar work with recommendations generated by the 'People You May Know' algorithm used in the LinkedIn social network, and, in 2016, Su et al. empirically and theoretically measured the effects of the Who-To-Follow system in the growth of Twitter.

The second line addresses the problem of influencing the growth of the network for obtaining some desired properties. In this perspective, we highlight the work by Parotsidis et al. (2016), who optimize the recommendation ranking to minimize the expected path length between the target user and the rest of the network.

In our work, we cover both lines: first, in chapters 4 and 5, we explore the effects of recommending users on the different properties of several Twitter graphs, providing a wider view than the cited articles; then, in chapter 6, we enhance some of those graph properties to evaluate the effects of those properties on the information diffusion over the network.

3. Recommendation Algorithms

Recommending contacts is a very recent functionality of online social networks (Hannon et al. 2010, Backstrom et al. 2011, Gupta et al. 2013): although they provided mechanisms for searching users, until the last few years of the previous decade, online sites such as Facebook, LinkedIn or Twitter did not provide any recommender system to find new users to follow. The development and comparison of contact recommendation algorithms is hence still a recent and largely open research field.

We provide in this chapter a thorough comparative revision of contact recommendation techniques reported in the literature, which we have selected for implementation and testing in our present research. The selected algorithms were originated in many different fields. We adapt and implement the most relevant and effective user recommendation and link prediction algorithms in a single evaluation framework, as well as further algorithms we propose here, based on adaptations of methods and models from the recommender systems and information retrieval fields. This section details the full collection of algorithms we shall use in the rest of this work.

3.1 Trivial Recommendation

First-off, as a rock-bottom reference and sanity-check in the evaluation of algorithms, we include two trivial recommendation approaches, namely random recommendation, and most-popular recommendation. Both are non-personalized recommendations, meaning that all users are recommended the same ranked list of contacts (except the relations that already exist for each user are excluded from the ranking for such user).

Random

Independently from the information contained in the network, this algorithm generates scores uniformly between 0 and 1 to create a completely randomized ranking.

$$f_u(v) = random(0,1) \tag{3.1}$$

This baseline is useful mainly to check for inconsistencies – an algorithm performing below random is a symptom of some error or anomaly – and as a measure of how easy or difficult the contact recommendation is in a particular dataset. The effectiveness of random recommendation is generally (and roughly speaking) proportional to the density of the social network, so this baseline provides a useful "zero bar" for a given dataset.

Popularity

Popularity-based recommendation ranks the different users according to the candidate contacts (i.e. users) according to their in-degree in the social graph.

$$f_u(v) = |\Gamma_{in}(u)| \tag{3.2}$$

Most-popular recommendation may seem at first sight like an oversimplified, not even personalized approach that one may expect to achieve a rather poor performance. However this approach is not really rare and is on the contrary used as an item recommendation functionality in a wide range of applications, such as news sites (most

read news), YouTube (most popular videos) or online stores (best-selling items). The reason for this extended use is that, on average, a user is most likely to consume an item that most people have previously consumed (Amatriain 2012). Recent studies have shown that most-popular recommendation achieves suboptimal but decent performance (Cremonesi et al. 2010), and we therefore include it as the true bottom baseline in our experiments.

3.2 Link Prediction Algorithms

The user recommendation problem we are addressing can be seen as a particular case (or a particular use) of the so-called link prediction problem in network science (Liben-Nowell 2003). The main difference between link prediction and user recommendation is the set of predictable links: the recommendation task restricts these sets to the edges that do not appear in the network and have their origin on a certain user (the target user of the recommendation).

Many algorithms have been proposed for the link prediction task in the last few years (Liben-Nowell 2003, Lü et al. 2011). We select here the most relevant methods, which we have adapted to the contact recommendation task. Link prediction algorithms can be classed into three categories: methods based on the neighborhoods of the endpoints of the possible edges, methods based on the paths which run through both endpoints, and methods based on random walks.

3.2.1 Neighborhood-based Methods

The link prediction algorithms within this category consider that the elements of the network that provide most information for predicting a new link are the sets of neighbors of both endpoints of the link. The similarity scores computed for this method are also known as local similarity indexes (Lü et al 2011), since it is not necessary to know the full network to compute them. The idea of using the neighborhoods of the nodes has given place to many algorithms, which we describe next.

Preferential Attachment

Barabási et al. (1999) observed that, in many real networks, new nodes connect themselves to nodes with high degree: this phenomenon is known as preferential attachment.

This algorithm tries to model this “rich gets richer” effect: The link prediction algorithm considers that a link between two nodes is most likely to appear when both endpoints of the graph have large degree (Liben-Nowell 2003). That principle motivates the following mathematical formulation for the score:

$$f_u(v) = |\Gamma(u)||\Gamma(v)| \tag{3.3}$$

Since the degree of the target user u is the same for all candidate users v in the contact recommendation task, we can drop the term $|\Gamma(u)|$, so the final expression simply becomes:

$$f_u(v) = |\Gamma(v)| \tag{3.4}$$

When the in-degree of the candidate nodes is considered, this algorithm is equivalent to the popularity-based recommendation as defined earlier.

Friend of a Friend (FOAF)

This method, proposed by Newman et al. (2001a) for predicting links in collaboration networks uses the number of common neighbors between the target and the candidate user to generate the corresponding ranking:

$$f_u(v) = |\Gamma(u) \cap \Gamma(v)| \tag{3.5}$$

In networks where the degree distribution is highly skewed, this algorithm tends to promote those nodes with the highest degree (since the probability that a node shares common friends with a lot of different nodes increases with the degree). Several approaches have been proposed to mitigate this effect. The following will be used in this work:

- **Jaccard coefficient:** This coefficient is commonly used as a similarity metric for documents in information retrieval systems (Salton et al. 1984). This coefficient measures the probability that two items x and y have a feature f for a randomly selected feature that either x or y has. Liben-Nowell et al. (2003) adapted the method for link prediction by taking the neighbors as the features, which leads to the following formula:

$$f_u(v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|} \tag{3.6}$$

- **Sørensen index:** Also known as the Dice coefficient, this index is mainly used for ecological community data (Sørensen, 1948). Lü et al. (2011) proposed its adaptation for link prediction:

$$f_u(v) = \frac{2|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u)| + |\Gamma(v)|} \tag{3.7}$$

- **Salton index:** This method applies a normalization inspired in cosine similarity (Salton et al. 1984). Lü et al. (2011) propose this approach to predict links by the following formulation:

$$f_u(v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{\sqrt{|\Gamma(u)||\Gamma(v)|}} \propto \frac{|\Gamma(u) \cap \Gamma(v)|}{\sqrt{|\Gamma(v)|}} \tag{3.8}$$

- **Leicht-Holme-Newman Index 1 (LHN1):** This index was originally proposed as a measure of structural equivalence of the nodes in networks (if they share many of the same network neighbors) (Leicht et al. 2006). It assigns high similarity to node pairs that have many neighbors in common in comparison to the expected number of common neighbors in a configuration model (Newman 2010, Ch. 13) for the degrees of the graph. It is computed as

$$f_u(v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u)||\Gamma(v)|} \propto \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(v)|} \tag{3.9}$$

  where the denominator is proportional to expected number of common neighbors in the configuration model. Lü et al. (2011) propose this method as a link prediction algorithm.
- **Hub Promoted Index (HPI) /Hub Depressed Index (HDI):** Ravasz et al. (2002) proposed the first of these indexes to compute the topological overlap of pairs of substrates in metabolic networks. Later, Zhou et al. (2009) redefined the index for predicting missing links in networks.

The ranking function for Hub Promoted Index is

$$f_u(v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{\min\{|\Gamma(u)|, |\Gamma(v)|\}} \tag{3.10}$$

while the one for Hub Depressed Index is

$$f_u(v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{\max\{|\Gamma(u)|, |\Gamma(v)|\}} \tag{3.11}$$

Although both approaches are very similar in their formulation, the nodes they promote in the recommendation are almost opposite: while Hub Promoted Index favors nodes with higher degree than the target user, Hub Depressed Index favors node with smaller degree than the target user.

Adamic-Adar

The score function for this algorithm was proposed as a similarity measure between users by Adamic et al. (2003). The principle is similar to FOAF: users with a highly overlapping friendship neighborhood are considered similar. But differently from FOAF, this measure gives more importance to common friends with a low degree (as being in a way more "unique" to both friendship circles) than to more popular people who have a lower discriminative power in comparing friendship neighborhoods.

$$sim(u, v) = \sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} \frac{1}{\log(freq(i))} \tag{3.12}$$

Taking the items as users, Liben-Nowell et al. (2003) adapted this measure to link prediction in the following way:

$$f_u(v) = \sum_{w \in (\Gamma(u) \cap \Gamma(v))} \frac{1}{\log|\Gamma(w)|} \tag{3.13}$$

Resource Allocation (RA)

This method (Zhou et al. 2009) is related to the physical process of resource allocation that takes place in social networks: Considering two nodes, u and v, the node u can send some resource to v. This resource has to be transmitted via the common neighbors of both users. First, user u transmits the resource to its neighbors. However, the resource cannot be duplicated, so each one of those neighbors cannot have a copy resource. User u, then, splits the resource in equal parts, and sends a part to each one of them. The neighbor nodes repeat the process with their own neighbors, and the fraction of the resource they have. Figure 8 shows an example of this process. After that second iteration, the amount of resource that v will receive from u will be:

$$f_u(v) = \frac{1}{|\Gamma(u)|} \sum_{w \in (\Gamma(u) \cap \Gamma(v))} \frac{1}{|\Gamma(w)|} \tag{3.14}$$

which is the similarity formulation for this coefficient. Figure 8 illustrates this process.

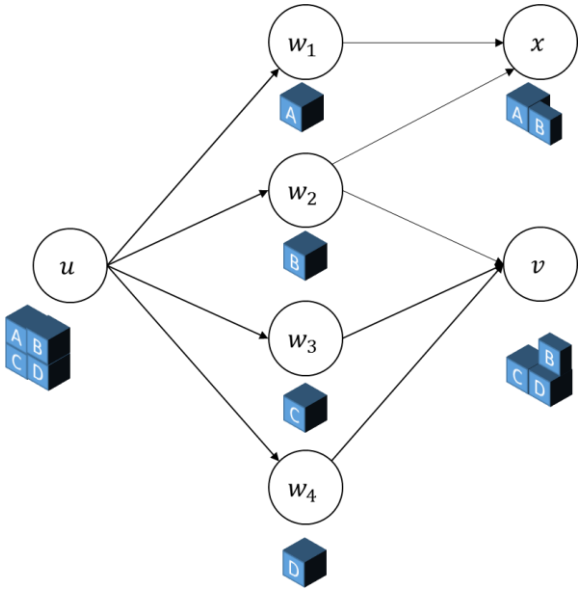


Figure 8. Resource allocation

Since the ranking is created for each user u separately, we can remove the first term and simplify the ranking function to:

$$f_u(v) = \sum_{w \in (\Gamma(u) \cap \Gamma(v))} \frac{1}{|\Gamma(w)|} \tag{3.15}$$

The main difference between this algorithm and Adamic-Adar is the absence of the logarithm: this algorithm penalizes more heavily the high degree of the neighbor nodes.

3.2.2 Path-based Methods

Expanding the vision of the recommendation algorithms to the full network allows taking advantage of several graph properties which are invisible at the local level, such as the existing paths between pairs of nodes in the network. We have selected and implemented the most important methods which use this information to improve the prediction.

Graph distance

This is the simplest distance based method of all, and consists on ranking the users by the length of the shortest path between them and the target user. Following the notion of small world network, in which individuals are related by short chains (Watts-Strogatz 1998), this algorithm uses the negative of the shortest path length as a score:

$$f_u(v) = -d(u, v) \tag{3.16}$$

where d is the minimum distance between u and v.

As this function takes only integer values, this method causes many draws between candidate contacts, which are resolved randomly.

Katz

This algorithm was proposed as a method for computing the status of a node in a social graph (Katz, 1953), and later proposed as a method for link prediction (Liben-Nowell, 2003). This method sums over the collection of all paths between the target node and

the candidate node, exponentially damped by the length of the path to give more importance to short paths.

The most general formulation is the following:

$$f_u(v) = \sum_{l=1}^{\infty} \beta^l |path^l(u \to v)| \tag{3.17}$$

where the expression $path^l(u \to v)$ represents the set of paths of length l between u and v. To ensure convergence of the infinite sum, the parameter β must be in the interval (0,1).

Since the sum contains infinite terms, there are two ways of computing it:

- The sum can be easily approximated, as β^l decreases quickly if we increase l. An accurate approximation to the real Katz matrix can be found in a few steps. This approach is taken by the method called **Local Path Index (LPI)** (Lü et al. 2009). Instead of using all path lengths, this algorithm uses only paths of length greater than 2. The original method used only paths of length 2 and 3, but it was later reformulated to allow paths of arbitrary length (Lü et al. 2011). This method can be expressed by the following formula:

$$f_u(v) = \sum_{l=0}^{n} \beta^l |path^{l+2}(u \to v)| \tag{3.18}$$

  where n+2 is the maximum length of the path.

- An exact expression can be found by using the properties of the adjacency matrix of the graph, A. The the n-th power of the adjacency matrix A^n has the number of paths of length n between two nodes as entries. This allows us to sum this as a geometric series, and obtain the following expression:

$$Katz(G) = \sum_{i=0}^{\infty} \beta^l A^l = (I - \beta A)^{-1} \tag{3.19}$$

  Once we have computed that matrix, computing the scores for every pair of users is straightforward:

$$f_u(v) = (I - \beta A)^{-1}_{uv} \tag{3.20}$$

Leicht-Holme-Newman Index 2 (LHN2)

This index (Leicht et al. 2006) was created for quantifying the similarity of vertices in networks. This similarity measure is based on the concept that two vertices are similar if their immediate neighbors in the network are themselves similar.

The starting point of this algorithm is a generalized version of the Katz formula, where each term A^l_{uv} has its own weight, C^l_{ij}. This weight is the inverse of the expected number of paths of length l between u and v in a configuration model:

$$\frac{1}{C^l_{ij}} = \mathbb{E}[A^l_{uv}] = \frac{|\Gamma(u)||\Gamma(v)|}{2m} \lambda_1^{l-1} \tag{3.21}$$

where m is the number of edges, and λ_1 is the greatest eigenvalue of the adjacency matrix. That leads to the following formulation:

$$f_u(v) = \frac{2m\lambda_1}{|\Gamma(u)||\Gamma(v)|}\left[\left(I - \frac{\phi}{\lambda_1}A\right)^{-1}\right]_{uv} \tag{3.22}$$

where ϕ is a free parameter.

Matrix Forest Index

This method depends on the Laplacian matrix of the network, which is computed as:

$$L = D - A \tag{3.23}$$

where A is the adjacency matrix of the graph, and D is the degree matrix of the graph, which is defined as:

$$D_{uv} = \begin{cases} |\Gamma(u)| & \text{if } u = v \\ 0 & \text{if } u \neq v \end{cases} \tag{3.24}$$

This matrix, also known as admittance matrix or Kirchhoff matrix, is a representation of the graph which may be used to find many useful properties of the graph, such as the number of spanning trees on it.

The Matrix Forest Index is defined as

$$(I - L)^{-1} \tag{3.25}$$

where L is the Laplacian matrix of the graph.

According to the Matrix-Forest theorem for directed multigraphs (Chebotarev et al. 1997), each coordinate of this matrix can be understood as the ratio of the number of spanning divergent forests such that the nodes u and v belong to the same divergent tree ($\mathcal{F}^{j\to i}$), rooted in u and the total number of spanning divergent forests for the network ($\mathcal{F}$). A divergent tree rooted in a vertex u is an acyclic connected directed subgraph such that any of its nodes can be accessed from node u. A spanning diverging forest is an acyclic subgraph of the network, all of whose components are diverging trees. As an example, Figure 9 shows the spanning divergent rooted forests for a 3-node cycle graph. Nodes in blue are the roots of each divergent tree.

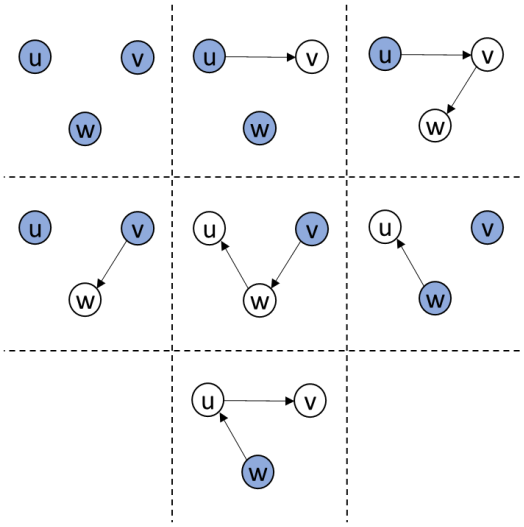


Figure 9. Spanning divergent forests for a 3-node cycle graph

Lü et al. (2011) proposed this method for link prediction, as well as a more general version, which adds a configurable parameter to the algorithm:

$$f_u(v) = (I + \alpha L)^{-1}_{uv} \tag{3.26}$$

where $\alpha > 0$.

3.2.3 Random Walk-based Methods

Social interactions can also be modeled by random walks. These methods use transition probabilities from a node to its neighbors to determine the destination of a random walker from a current node.

PageRank

PageRank (Brin et al. 1998) exploits the network link structure to generate scores for the different nodes. The score for a single node, u, represents the stationary probability of being in that node for a user who 'walks' randomly through the network edges. Originally proposed for ranking web pages depending on the hyperlink topology, it is one of the most well-known random walk algorithms.

The importance of a node relies on three factors: the number of nodes that point to it, their importance, and the number of outgoing edges from those nodes. If a node is linked to by a high amount of nodes, the importance of that node increases. The importance transmitted by an in-link is recursively proportional to the importance of the link source. Moreover, the importance transmitted by the link is decreased proportionally to the number of outgoing edges from the source. This leads to the following recursive equation:

$$p(v) = \frac{r}{|U|} + (1-r) \sum_{w \in \Gamma_{\text{in}}(v)} \frac{p(w)}{|\Gamma_{out}(w)|} + \frac{(1-r)}{|U|} \sum_{w:|\Gamma_{out}(w)|=0} p(w) \tag{3.27}$$

where r is a parameter that represents the probability that the random walker teleports to a random node (ignoring links). The rightmost term in the equation helps to handle the sinks in the network, ensuring that that $\sum_w p(w) = 1$.

The values for each node are iteratively computed, using the expression above. The initial weights for each node are

$$p_0(v) = \frac{1}{|U|} \tag{3.28}$$

Once the algorithm has converged, the score is simply generated as

$$f_u(v) = p(v) \tag{3.29}$$

As it can be observed, we do not use information about who the target user is to generate the scores for this algorithm, so the recommendation is the same for every user. Applying a few modifications to the algorithm, it is possible to obtain a personalized version. Many of these versions have been documented. Between all of them, we shall use the most common, proposed by White & Smith (2003).

The modification lies on the teleport probability. In the basic version, every user could teleport to any node in the network with the same probability. In the personalized version, the user can only teleport to the target user. By doing that, the importance of a node does not only depend on the link structure of the network, but also on how near the node is to the target user. The formula is the following:

$$p_u(v) = r\delta_{uv} + (1-r)\sum_{w\in\Gamma_{in}(d_j)}\frac{p_u(w)}{|\Gamma_{out}(w)|} + (1-r)\delta_{uv}\sum_{w:|\Gamma_{out}(w)|=0} p_u(w) \tag{3.30}$$

where δ_{xy} is a Kronecker delta function, and u is the target user. This algorithm has been used in link prediction under the name of **rooted PageRank** (Liben-Nowell et al. 2003), since the target user acts as a 'root' of the random walk.

Additionally, we propose a new variant of this personalized algorithm. This variant**, pure personalized PageRank**, assumes that the target vertex can only be reached by teleportation. To do that, it is enough to eliminate all links from nodes to the target user. When the random walker reaches a sink, it teleports with probability r to the target user, or continues the random walk teleporting to a different node with probability $(1-r)$. By doing that, the possible bias introduced by the differences of the target nodes neighborhoods is removed.

$$f_u(v) = ppp_u(v) = \begin{cases} r & \text{if } v = u \\ (1-r)\sum_{w\in\Gamma_{in}(v)}\frac{p_u(w)}{|\Gamma_{out}(w)-\{u\}|} + \frac{(1-r)}{N-1}\sum_{w:|\Gamma_{out}(w)-\{u\}|=0} p_u(w) & \text{if } v \neq u \end{cases} \tag{3.31}$$

The initialization of the users for the algorithm has also been changed to:

$$ppp_0(u) = \begin{cases} r & \text{if } u \text{ is the target user} \\ \frac{1-r}{|U|-1} & \text{if not} \end{cases} \tag{3.32}$$

A derivation of this last formula can be seen in Annex I: Derivations.

Hitting time

Also known as mean first passage time, the hitting time from a node u to a node v is the expected number of steps required for a random walk starting at u to reach node v:

$$H(u,v) = \sum_{t=0}^{\infty} t\left(p_{u\to v}(t) - p_{u\to v}(t-1)\right) \tag{3.33}$$

where $p_{u\to v}(t)$ represents the probability of reaching node v starting from node u in time smaller or equal to t. This probability is represented by a transition matrix T, defined as follows:

$$T_{uv} = \begin{cases} \frac{\lambda}{|\mathcal{U}|} + (1-\lambda)\frac{A_{uv}}{|\Gamma_{out}(u)|} & \text{if } |\Gamma_{out}(u)| > 0 \\ \frac{1}{|\mathcal{U}|} & \text{if } |\Gamma_{out}(u)| = 0 \end{cases} \tag{3.34}$$

The final score for this algorithm is the additive inverse of $H(u,v)$: this algorithm finds the easiest nodes to reach with a random walk.

$$f_u(v) = -H(u,v) \tag{3.35}$$

For computing the matrix H, we will use the algorithm proposed by Meyer (1975). This algorithm defines H as:

$$M = \left[I - B^{\#} + JB_{dg}^{\#}\right]\Pi^{-1} \tag{3.36}$$

where $B = I - T$, and $B^{\#}$ is the pseudoinverse matrix of B. B_{dg} represents the matrix B with 0 in the main diagonal, and Π is a diagonal matrix whose entries represent the stationary probability of each node.

In addition to the mean first passage time, another related measure is used for predicting links in a network. This new measure, called **commute time** represents the expected time for the random walker to travel from u to v and returning from v to u (Liben-Nowell 2003).

$$f_u(v) = -H(u,v) - H(v,u) \tag{3.37}$$

PropFlow

The PropFlow algorithm (Lichtenwalter et al. 2010) computes the probability that a restricted random walk starting at u ends at v in l steps or fewer using link weights as transition probabilities. In case that the graph is unweighted, it is considered that all links have weight equal to 1. The walk terminates upon reaching v or upon revisiting any node (including u). The random walk selects links based on their weights, which produces a score that can be used as an estimation of the likelihood of new links.

$$f_u(v) = \text{propflow}_u(v) \tag{3.38}$$

The steps to compute this score are shown in detail in Algorithm 1

Algorithm 1. PropFlow

for $v \in \mathcal{U}$
    $\text{propflow}_u(v) = 0$
end for
$\text{propflow}_u(u) = 1$
$found \leftarrow \{u\}$
$newSearch \leftarrow \{u\}$
for $i = 1 \ldots l$
    $oldSearch \leftarrow newSearch$
    $newSearch \leftarrow \{\}$
    while $(!\, oldSearch.isEmpty())$
        $x \leftarrow oldSearch.pop()$
        for $v \in \Gamma(x)$
            $\text{propflow}_u(v) \leftarrow \text{propflow}_u(x) \cdot \frac{weight(x,v)}{\sum_{y \in \Gamma(x)} weight(x,y)}$
            if$(v \notin found)$
                $found \leftarrow found \cup \{v\}$
                $newSearch \leftarrow newSearch \cup \{v\}$
        end for
    end while
end for
return propflow_u

PropFlow is similar to rooted PageRank, but it is cheaper to compute, since only a breadth first search with maximum depth l is needed. This algorithm is also insensitive to topological noise far from the source node.

3.3 Twitter Who-To-Follow

Twitter is, nowadays, one of the largest massive online social networks. In 2010, the Who-To-Follow service[3] was deployed by Twitter into its online platform, which aimed to provide personalized user recommendations. To generate those recommendations, several methods were developed, the fundamentals of which Twitter disclosed (Gupta et al. 2013, Goel et al. 2015). In this section, we will discuss those algorithms.

Due to the massive scale of the full Twitter graph, it is infeasible for modern technology to generate recommendations for every single user by the analysis of the entire graph in an affordable period of time. To avoid that, the Who-To-Follow service builds a bipartite graph known as the consumer-producer graph is computed for each user in the network. The "left side" of the graph (link sources), known as the set of consumers or hubs of the graph, contains a so-called circle of trust of the target user. The "right side" of the graph (link destinations), known as the set of producers or authorities, is formed by every user followed by a node in the circle of trust. An example of this graph is illustrated on Figure 10.

The circle of trust for a user is built as a subset of k vertices of the social network recovered by an "egocentric" random walk, very similar to personalized PageRank (Gupta et al. 2013). In case that the number k of elements we want to select for the circle of trust is greater than the actual number of nodes in the graph, every user is included.

The detailed steps for computing the bipartite graph are as the follows:

1. Compute the user's circle of trust by running the egocentric random walk on the graph. The scores for this random walk are computed using the personalized PageRank algorithm. Add the top k items to the circle of trust. Add the target node to this circle of trust.
2. Take all the outgoing edges from the circle of trust, add the adjacent nodes to the bipartite graph, as producers, and add the links from the consumers to the producers. If a producer was in the circle of trust, it is duplicated and added to the producers set all the same.

An option to further alleviate scalability, used by Twitter team for some algorithms, consists on subsampling a local neighborhood of the target user, and generating the bipartite graph using only those users (Goel et al. 2015). In our experiments, we shall nonetheless use the full bipartite graph construction.

[3] https://blog.twitter.com/2010/discovering-who-to-follow (Last access: 11/11/2016)

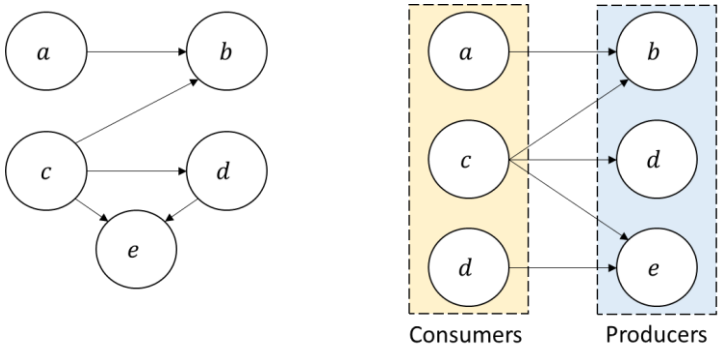


Figure 10. Consumer-Producer graph (adapted from Goel et al. 2015)

On the consumer-producer graph, different recommendation algorithms can be applied. Three methods have been explored by Twitter's researchers: HITS, SALSA and a variant of cosine similarity.

HITS

HITS is an acronym from Hypertext-Induced Topic Search (Kleinberg 1999B). This algorithm defines and computes scores over the nodes of the network, which can be used as a ranking function to recommend contacts. This algorithm was originally created for using the link structure of the Web to obtain better search results. The main idea behind this algorithm is the following: hub nodes and authority nodes maintain what is called a mutually reinforcement relationship: a good hub is a node that points to many good authorities, and a good authority is a node that is pointed by many good hubs. Formally, let $h(v)$ and $a(v)$ be, respectively, the hub and authority values for a user v. These values are defined so that the following equations are satisfied for all users:

$$\begin{aligned} a(v) &= \sum_{w \in \Gamma_{in}(v)} h(w) \\ h(v) &= \sum_{w \in \Gamma_{out}(v)} a(w) \end{aligned} \tag{3.39}$$

where $h(v)$ and $a(v)$ are normalized (the sum of the squares of each measure is set to one):

$$\sum_{u} a^2(u) = \sum_{u} h^2(u) = 1 \tag{3.40}$$

Similar to PageRank, this algorithm is computed iteratively, alternating between hub and authority value updates.

This algorithm is obviously not personalized: it depends nowhere of a target user. Several approaches have been documented for personalizing this algorithm (White & Smith, 2003). As proposed by Twitter, we will use a teleport vector to the hubs score for personalizing this algorithm, in the following way:

$$\begin{aligned} f_u^h(v) = h_u(v) &= \alpha \delta_{uv} + (1-\alpha) \sum_{w \in \Gamma_{out}(v)} a(w) \\ f_u^a(v) = a_u(v) &= \sum_{w \in \Gamma_{in}(v)} h(w) \end{aligned} \tag{3.41}$$

$$\sum_{u \in \mathcal{U}} a^2(u) = 1, \sum_{u \in \mathcal{U}} h^2(u) = 1$$

When this algorithm is applied over the circle of trust, it is known as **Love** (Goel 2014). That is because, since love is intangible, it can be fully given to all people you know without running out of it. This is similar to how HITS spreads both types of scores: every node in the bipartite graph directly receives its neighbors full scores.

As we are already working with reduced samples of Twitter, we shall use bipartite graphs with different sizes for the circle of trust, and the complete bipartite graph of the network (as if the number of users in the circle of trust is infinite). This last version will be called, from now on, **Personalized HITS**, to differentiate it from the Twitter version. In all the versions, both hub scores $f_u^h(v)$ and authority scores $f_u^a(v)$ will be used for recommending users.

SALSA

SALSA (Lempel et al. 2001) stands for Stochastic Approach for Link-Structure Analysis. Similar to PageRank and HITS, SALSA was originally proposed to take advantage of the link-structure of the Web to obtain better results in search engines.

SALSA analyzes two random walks at a time: a random walk on the authorities, and a random walk on the hubs. The walks are made of double steps, where state transitions correspond to traversing two links in a row: one link forward and one link backward, or viceversa. In this update, if the random walker has d distance 1 neighboring nodes to which he can transition, a uniform a fraction of the score $1/d$ is transferred to each one of those nodes. The scores are computed as

$$\begin{aligned} f_u^a(v) &= a(v) \\ f_u^h(v) &= h(v) \end{aligned} \tag{3.42}$$

where $a(u)$ represents the authority score for user u, $h(u)$ represents the hub score for the hub, and the scores are defined as follows:

$$\begin{aligned} a(u) &= \sum_{w \in \Gamma_{in}(u)} \frac{h(w)}{|\Gamma_{out}(w)|} = \sum_{w \in \Gamma_{in}(u)} \sum_{v \in \Gamma_{out}(w)} \frac{a(v)}{|\Gamma_{in}(w)||\Gamma_{out}(v)|} \\ h(u) &= \sum_{w \in \Gamma_{out}(u)} \frac{a(w)}{|\Gamma_{in}(w)|} = \sum_{w \in \Gamma_{out}(u)} \sum_{v \in \Gamma_{in}(w)} \frac{h(v)}{|\Gamma_{out}(w)||\Gamma_{in}(v)|} \end{aligned} \tag{3.43}$$

Simplifying the previous equations, it is possible to obtain the following formulas for the hub and authority score of each user.

$$\begin{aligned} a(u) &= \frac{|A_{c(u)}|}{|A|} \frac{|\Gamma_{in}(u)|}{\mathcal{W}_{c(u)}} \\ h(u) &= \frac{|H_{c(u)}|}{|H|} \frac{|\Gamma_{out}(u)|}{\mathcal{W}_{c(u)}} \end{aligned} \tag{3.44}$$

where $c(u)$ represents the strongly connected component of the graph which contains vertex u, H is the set of hubs, A is the set of authorities and

$$\mathcal{W}_c = \sum_{w \in c} |\Gamma_{in}(w)| = \sum_{w \in c} |\Gamma_{out}(w)| \tag{3.45}$$

For contact recommendation, we can remove $|H_{c(u)}|$, $|A_{c(u)}|$ and $|\mathcal{W}_{c(u)}|$ from the formulas, since they do not depend on the candidate user, and also, $|A|$ and $|H|$ since they are constants for all the users in the bipartite graph. As a result, we obtain that the authorities scores for SALSA are equivalent to the most-popular recommendation ones, and the hubs scores are equivalent to choosing the outgoing neighborhood in the Preferential Attachment algorithm.

A more interesting version of SALSA is the personalized version which can be defined similarly to HITS and PageRank: adding a user-centered teleport vector to the hubs, as follows:

$$f_u^h(v) = h_u(v) = \alpha\delta_{uv} + (1-\alpha)\sum_{w\in\Gamma_{out}(v)}\frac{a(w)}{|\Gamma_{in}(v)|}$$
$$f_u^a(v) = a_u(v) = \sum_{w\in\Gamma_{in}(v)}\frac{h(w)}{|\Gamma_{out}(w)|} \tag{3.46}$$

Over Love and the cosine similarity variant we will explain later, this version of the algorithm corresponds to the contact recommendation algorithm that Twitter's researchers selected for the implementation of the Who-To-Follow system, and it receives the name of **Money** (Goel 2014, Goel et al. 2015). Money, in contrast with love, is finite, so it has to be divided between all the people who receives it. That is how SALSA acts: it shares a certain score equally between all the neighbors of a node in the bipartite graph.

Over the bipartite graph, scores are computed iteratively using the formulas above. The original approach documented in the literature (Goel et al. 2015) proposes using the authority scores to produce recommendations, though we could also recommend the hubs. In our experiments, we will study both options.

In the same as HITS, this algorithm can be run on the reduced circles of trust or over the full bipartite graph. We shall call **Personalized SALSA** the version over the complete bipartite graph.

Cosine Similarity

In addition to Love and Money, Goel et al. (2015) studied another method for their system: this method recommends users who have a high cosine similarity with the users that the target user follows (Goel et al. 2015). The vectors to use in the similarity calculations will only be computed for each one of the nodes in the authorities' side. The dimension of the vectors is the number of hubs in the bipartite graph, and, for each one of them, the j-th coordinate is 1 if the j-th hub has a link to the authority, and 0 otherwise. We denote the vector for each authority as $\vec{v}_{in}$.

$$(\vec{v}_{in})_x = \begin{cases} 1 & \text{if } (x,v) \in E \\ 0 & \text{if } (x,v) \notin \text{E} \end{cases} \tag{3.47}$$

It is not fully known how Twitter uses the cosine similarity to compute the scores, since they have only provided an example of the algorithm over a network (Goel et al. 2014, Goel et al. 2015). We propose three different possibilities for that algorithm which fulfil that example:

- **Centroid cosine similarity:** This method computes a centroid for the user, $\vec{u}_{out}$, using the vectors of the followed users. The score by this method is:

$$f_u(v) = \cos(\vec{u}_{out}, \vec{v}_{in}) = \frac{\sum_w (\vec{u}_{out})_w (\vec{v}_{in})_w}{|\vec{u}_{out}||\vec{v}_{in}|} \propto \frac{\sum_w (\vec{u}_{out})_w (\vec{v}_{in})_w}{|\vec{v}_{in}|} = \frac{\sum_w (\vec{u}_{out})_w (\vec{v}_{in})_w}{\sqrt{|\Gamma_{in}(v)|}} \quad (3.48)$$

where $\vec{u}_{out}$ is the average of the vectors generated for each adjacent producers of node u in the bipartite graph:

$$\vec{u}_{out} = \frac{1}{|\Gamma_{out}(u)|} \sum_{w \in \Gamma_{out}(u)} \vec{w}_{in} \quad (3.49)$$

- **Average cosine similarity:** This method computes all the similarities between the authorities, and scores recommended contacts by the average similarity over the authorities that the target user is currently following.

$$f_u(v) = \frac{1}{|\Gamma_{out}(u)|} \sum_{w \in \Gamma_{out}(u)} \cos(\vec{w}_{in}, \vec{v}_{in}) \quad (3.50)$$

Since the denominator term is the same for all the candidate users, it can be removed from the formula. The final equation is:

$$f_u(v) = \sum_{w \in \Gamma_{out}(u)} \cos(\vec{w}_{in}, \vec{v}_{in}) = \sum_{w \in \Gamma_{out}(u)} \frac{|\Gamma_{in}(w) \cap \Gamma_{in}(v)|}{\sqrt{|\Gamma_{in}(w)||\Gamma_{in}(v)|}} \quad (3.51)$$

- **Maximum cosine similarity:** This method is similar to the previous one but computes the maximum rather than the average similarity over the friends of the target user.

$$f_u(v) = f(u, v) = \max_{w \in \Gamma_{out}(u)} \cos(\vec{w}_{in}, \vec{v}_{in}) = \max_{w \in \Gamma_{out}(u)} \frac{|\Gamma_{in}(w) \cap \Gamma_{in}(v)|}{\sqrt{|\Gamma_{in}(w)||\Gamma_{in}(v)|}} \quad (3.52)$$

3.4 Recommender System Methods

Beyond the algorithms reported here so far, which directly proposed for the problem of recommending contacts in social networks, it seems natural to consider whether methods originally devised in the recommender systems fields to recommend items (news, movies, books, clothes, and other goods) with a domain-independent perspective could be adapted and be effective at recommending people. The social network adjacency matrix can be seen as a (binary of weighted, depending on the nature of the graph) user-item rating matrix after all, with the interesting particularity that items and users are the same space, whereupon it is immediate to apply these algorithms to social network, in such a way that the predicted unknown matrix cell values (or scores) can be interpreted as a contact recommendation (the "column" candidate user) to a ("row") target user.

It is one of the specific goals of our research to explore this direction and compare the effectiveness of this adaptation to such other alternatives as we have described in the

previous sections. From the extensive literature of recommender systems, we focus on the most representative, which we describe next.

3.4.1 K-Nearest Neighbors

Nearest neighbors algorithms (kNN) are among the most used and widely known collaborative filtering approaches. These recommenders rely on the principle that similar users prefer similar items, and similar items are preferred by similar users (Ning et al 2015). These methods select the top k most similar neighbors to the target user or item, and compute the recommendation score for them as a linear combination of the neighborhood ratings.

kNN algorithms are very configurable: the similarity function between users or items, the size of the neighborhood, the normalizations, if any, or how to deal with exceptions related to lack of sufficient data, are some of the settings to be the defined in the methods. One of the top-level options to decide on is whether to work with user or item neighborhoods.

User-based k nearest neighbors

User-based nearest neighbors algorithms compute the score of a candidate item i for a target user u using the ratings that most similar users to u have assigned to i. This group of users is known as the neighborhood of u and the score is defined as follows:

$$r_u(i) = \sum_{v \in \mathcal{N}(u)} sim(u, v) r_v(i) \tag{3.53}$$

where $sim(u, v)$ represents the similarity between u and v, and $\mathcal{N}(u)$ is the neighborhood of u. In the adaptation of the method for contact recommendation, $r_v(i)$ is equal to 1 when the link between v and i exists, and 0 if not, so the formula is rewritten as:

$$f_u(v) = \sum_{\substack{w \in \mathcal{N}(u) \\ (w,v) \in E}} sim(u, w) \tag{3.54}$$

The similarity between users can be assessed in different ways. In the collaborative filtering approach, the similarity between users should be defined based on their interactions with items, which in our context translates to their social connections, i.e. their outgoing links. For instance, using the cosine similarity:

$$sim(u, v) = \frac{|\Gamma_{\text{out}}(u) \cap \Gamma_{\text{out}}(v)|}{\sqrt{|\Gamma_{\text{out}}(u)||\Gamma_{\text{out}}(v)|}} \tag{3.55}$$

Other typical similarity functions can be used in place of cosine, such as the Jaccard similarity, Pearson correlation, etc.

Item-based k nearest neighbors

Item-based nearest neighbor algorithms predict the score of an item i for a user u based on the ratings u has provided to the most similar items to i. The score is defined as:

$$r_u(i) = \sum_{j \in \mathcal{N}(i)} r_u(j) sim(i, v) \tag{3.56}$$

Analogously to the user-based version, in the domain of contact recommendation, the ranking function translates to:

$$f_u(v) = \sum_{\substack{w \in \mathcal{N}(v) \\ (u,w) \in E}} sim(v, w) \tag{3.57}$$

Since in contact recommendation users and items are the same space, the item-based similarity can use the same functions as user-based. The only difference is the direction of the neighborhood: item-based kNN should use the incoming neighbors $\Gamma_{in}(u)$, $\Gamma_{in}(v)$ instead of Γ_{out} in the equation for cosine similarity.

3.4.2 Matrix Factorization

By the mid of the past decade, a new collaborative filtering approach started to show excellent performance in the recommendation task, based on factorizing the user-item rating matrix with respect to a space of latent factors. Matrix factorization defines a family of model-based collaborative filtering algorithms, which approximate the rating matrix as a product of two matrices: one matrix for the users, $X \in \mathbb{R}^{|\mathcal{U}| \times k}$, and one for the items, $Y \in \mathbb{R}^{|\mathcal{I}| \times k}$. Each row of those matrices represents a user or an item in a joint latent factor space of dimension k, typically much smaller than the number of items or users (Koren et al. 2009).

For the items, each latent variable represents a characteristic for the items. For example, in the case of movies, one latent variable could represent the amount of action of the film. For users, the elements of the vector measure to what extent the user is interested in items which have high values in the corresponding factor.

We will describe next two of the most representative, effective and popular matrix factorization algorithms for collaborative filtering.

Matrix Factorization for Implicit Feedback

The earliest matrix factorization algorithms took as input explicit user rating values (Sarwar et al. 2000). Later, Hu et al. (2008) devised an algorithm specifically aimed at using implicit user feedback, where the user-item interactions observed by the system may not necessarily rating values explicitly entered by users for items, but a record of a natural interaction, such as clicking, playing, buying, etc. For example, implicit feedback for a music track could be how many times a user has fully played it. Hence the goal of this algorithm is not predicting the rating value would assign the item but predicting whether the user would consume the item.

For such purpose, Hu et al. compute for each user-item pair two values which depend on the implicit feedback: a preference value, p_{ui}, which we aim to predict, and a confidence level c_{ui} for that preference value. The preference value can be obtained by simply binarizing the implicit feedback value associated to the user-item interaction:

$$p_{ui} = \begin{cases} 1 & \text{if } r_{ui} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.58}$$

where r_{ui} is the original feedback value. The confidence value is a monotonous function of the implicit feedback value. In the original paper, the following function is proposed:

$$c_{ui} = 1 + \alpha r_{ui} \tag{3.59}$$

where $\alpha > 0$ is a free parameter. Following with the music example, if a user has played a music track many times, it seems quite likely that he enjoys the track.

This matrix factorization technique tries to factorize the binarized preference matrix in two separate matrices: one for the users, $X \in \mathbb{R}^{|\mathcal{U}| \times k}$, and another one for the items

$Y \in \mathbb{R}^{|\mathcal{I}| \times k}$. Each user and each item is represented by a vector in the latent space model, and the predicted preference value for that object is computed as their inner product:

$$f_u(v) = x_u^T y_i \tag{3.60}$$

Hu et al. propose computing the latent space vectors by minimizing the following objective function:

$$\min_{x*,y*} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right) \tag{3.61}$$

The $\lambda(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2)$ term is necessary for regularizing the model and prevent overfitting. λ is a free parameter, greater than zero.

In our particular case, where items are the same as the users, the implicit feedback value will be the weight of the edge between the target and the candidate user (which in the simplest case is 1 if the edge exists, and 0 if it does not). In the rest of the present document, we shall refer to this method as ImplicitMF.

Probabilistic Matrix Factorization

As a method which uses the explicit feedback provided for the users, we have selected one of the most relevant methods, Probabilistic Matrix Factorization, which aims to minimize RMSE error of a recommender (Salakhutdinov et al. 2007). For doing that, this method first defines a linear probabilistic graphical model with Gaussian observation noise, as shown in Figure 11, which defines the following conditional distribution over the observed ratings:

$$p(R|X, Y, \sigma^2) = \prod_{u=1}^{|\mathcal{U}|} \prod_{i=1}^{|\mathcal{I}|} \left[\mathcal{N}\left(r_u(i) \middle| x_i^T y_j, \sigma^2\right) \right]^{I_{ui}} \tag{3.62}$$

where $\mathcal{N}(x|\mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and standard deviation σ, and

$$I_{uv} = \begin{cases} 1 & \text{if } r_u(i) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.63}$$

This model considers that the ratings are normally distributed around the inner product of two latent vectors: $x_u, y_i \in \mathbb{R}^k$. Then, the rating is computed as

$$f_u(v) = x_u^T y_i \tag{3.64}$$

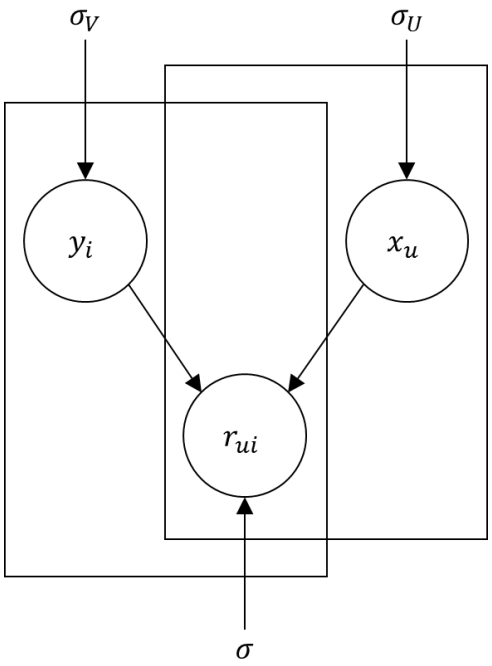


Figure 11. Probabilistic Matrix Factorization Graphical Model (adapted from Salakhutdinov & Minh 2007)

Assuming that the priors of both user and item vectors follow zero-mean spherical Gaussian distributions, we get:

$$p(X|\sigma_U^2) = \prod_{u=1}^{|\mathcal{U}|} \mathcal{N}(x_u|0, \sigma_U^2 I) \quad p(Y|\sigma_I^2) = \prod_{i=1}^{|\mathcal{I}|} \mathcal{N}(y_i|0, \sigma_I^2 I) \tag{3.65}$$

We want to maximize the log-posterior distribution over both the user and item features (the latent variables). Fixing the noise variance and the prior variances, that maximization is equivalent to minimizing the following function:

$$\min_{x*,y*} \sum_{u,i} I_{ui}(r_u(i) - x_u^T y_i)^2 + \lambda_U \sum_u \|x_u\|^2 + \lambda_I \sum_i \|y_i\|^2 \tag{3.66}$$

where $\lambda_U = \sigma^2/\sigma_U^2$ and $\lambda_I = \sigma^2/\sigma_I^2$ are free parameters of the model. The minimum can be found, for instance, by a gradient descent in X and Y.

The previous formulation can yield ratings values outside the allowed range. To ensure that the predicted ratings are in range (in our case, between 0 and 1), it is possible to compute the ratings as a function of the inner product. For example, it is possible to apply a sigmoid function:

$$f_u(v) = S(x_u^t y_i) \tag{3.67}$$

where

$$S(x) = \frac{1}{1 + e^{-x}} \tag{3.68}$$

In this case, the objective function to optimize is the following:

$$\min_{x*,y*} \sum_{u,i} I_{ui}\left(r_u(i) - S(x_u^t y_i)\right)^2 + \lambda_U \sum_u \|x_u\|^2 + \lambda_I \sum_i \|y_i\|^2 \tag{3.69}$$

To distinguish both versions, we shall refer to the first version as **basic probabilistic matrix factorization** (PMFBasic), and to the second as **sigmoid probabilistic matrix factorization** (PMFSigmoid).

3.5 Text Information Retrieval Methods

Given a query issued by a user, classic text information retrieval (IR) models seek to obtain the most relevant documents for the information need expressed by the query, based on an analysis of the words in the documents and the words in the query. Recommender systems can be seen as a quite distinct particular case of information retrieval where the query is empty, or not explicit, and records of user activity are available instead. However, it is still possible to apply IR to recommendation, by mapping the variables involved in the recommendation problem (users, items, interactions between them) to the IR task spaces (queries, documents, words).

The mapping can be defined in different ways. In user recommendation, users can play three roles: Candidate users can be seen as the documents to retrieve, while the target user is equated to the query. Both elements are represented by their neighbors, who can be the equivalent of words in text IR.

The relationship between text retrieval methods and recommender systems has been explored by some authors (Bellogín et al. 2013, Hannon et al. 2010), but there is still work to be done in this direction. In our research, we have adapted several text information retrieval methods. Since it poses a novel way to recommend contacts, one of the main goals of our experiments consists on comparing the effectiveness of these algorithms with the rest of proposed approaches.

The methods we have adapted are the most relevant and well-known methods documented in the IR literature. Those algorithms are known as TF-IDF model, BIR and BM-25 probabilistic models and Query Likelihood language model. In the rest of this section, those methods are described and an adaptation for the contact recommendation problem is proposed.

TF-IDF

One of the most well-known IR models is known as the vector model (Salton et al. 1975). In this model, documents and queries are represented as $|\mathcal{V}|$-dimensional vectors, where $\mathcal{V}$ represents the vocabulary of the collection (the set of words included in, at least, one document in the collection).

Each term in a document is assigned a certain weight, which defines the relevance of that term for the given document (Baeza-Yates et al. 2011). This weight is computed as the product of two separate terms: the term frequency (tf) and the inverse document frequency (idf). As the number of occurrences of a word in a document increases, so does the term frequency. It is assumed that high frequency words in a document are important for describing the topic. Inverse document frequency measures the discriminative power of the term: if a term appears in a few documents, it will identify them better than another one that appears in most part of the collection. The formulas for these terms are:

$$\mathrm{tf}(w,d) = \begin{cases} 1 + \log_2 freq(w,d) & \text{if } freq(w,d) > 0 \\ 0 & \text{if } freq(w,d) = 0 \end{cases} \tag{3.70}$$

$$\text{idf}(w) = \log_2\left(1 + \frac{|\mathcal{D}|}{1 + |\mathcal{D}_w|}\right) \tag{3.71}$$

where $\mathcal{D}$ is the full collection of documents, $\mathcal{D}_w$ is the set of documents in the collection which contain term w, and $freq(w, d)$ is the number of appeareances of term w in the document d.

The tf-idf weight is thus associated to a term and a document. To recover a document in response to a query, it is necessary to compute the degree of similarity between each document and the corresponding query. To do that, cosine similarity is applied:

$$sim(d, q) = \cos(d, q) = \frac{\sum_{w \in d \cap q} \text{tf-idf}(w, d)\text{tf-idf}(w, q)}{\sqrt{\sum_{w \in d} \text{tf-idf}(w, d)^2}\sqrt{\sum_{w \in q} \text{tf-idf}(w, d)^2}} \tag{3.72}$$

The response to a query by the Vector Similarity Model (VSM) consists of a ranked list of documents ordered by descending similarity to the query. As the ranking is unaffected by the module of the query vector (it is constant for every document), it is possible to eliminate it from the equation.

The tf-idf VSM can be adapted to contact recommendation by mapping the triad (query, term, document) to the triad (target user, adjacent user, candidate user), as follows:

$$\text{tf-idf}(w, u) = \text{tf}(w, u) \cdot \text{idf}(w) \tag{3.73}$$

$$\text{tf}(w, u) = \begin{cases} 1 & \text{if } w \in \Gamma(u) \\ 0 & \text{otherwise} \end{cases} \tag{3.74}$$

$$\text{idf}(w) = \log_2\left(1 + \frac{|\mathcal{U}|}{1 + |\Gamma(w)|}\right) \tag{3.75}$$

$$f_u(v) = \frac{\sum_{w \in \Gamma(u) \cap \Gamma(v)} \text{tf-idf}(w, u) \cdot \text{tf-idf}(w, v)}{\sqrt{\sum_{w \in \Gamma(v)} \text{tf-idf}(w, v)^2}} \tag{3.76}$$

In both tf and idf equations, we may take the incident or the adjacent links of both target and candidate users.

Binary Independent Retrieval (BIR)

Binary Independent Retrieval (Robertson & Sparck Jones 1976) is the one of the earliest and most basic probabilistic models in information retrieval. This algorithm follows the Probability Ranking Principle (Robertson 1977): Given a query, a document is ranked by decreasing order of the probability of relevance of the corresponding document for the query. That probability is computed in this method using only the document representation (the information that is available to the system).

BIR considers that the frequency distribution of the terms in a document follows a Bernouilli distribution, and its general formula is:

$$sim(d, q) = \sum_{w \in \text{d} \cap \text{q}} RSJ(w) \tag{3.77}$$

where $RSJ(w)$ is the Robertson-Sparck Jones formula, which is defined as

$$RSJ(w) = \log\left(\frac{|R_w|(|\mathcal{D}| - |\mathcal{D}_w| - |R| + |R_w|)}{(|R| - |R_w|)(|D_w| - |R_w|)}\right) \tag{3.78}$$

where R is the set of relevant documents for the query, R_w is the set of relevant documents which contain the term w, $\mathcal{D}$ is the full document collection, and $\mathcal{D}_w$ is the set of documents that contain w.

Most of the times, there is no a priori information about the relevance of the documents, so it would be impossible to apply this formula literally. An approximation is used instead, based on the consideration that typically only a tiny fraction of all documents are relevant for a common query:

$$RSJ'(w) = \log\left(\frac{|\mathcal{D}| - |\mathcal{D}_w| + 0.5}{|D_w| + 0.5}\right) \tag{3.79}$$

This function has clear similarity to the inverse document frequency of a document: if a term appears in most of the collection, then, its weight for computing the relevance of the document will be very small, while more discriminative terms obtain more importance. Note however that the ranking function disregards the frequency of words in documents, that is, it just takes into account the presence or absence of terms in documents (hence the name "binary"). As a consequence, this model is by itself quite ineffective at the search task compared to other methods that do pay attention to word frequency. The method is nonetheless useful as a component of other more elaborate approaches, an example of which, BM25, we will describe next.

Despite its suboptimality as a stand-alone IR model, we shall explore the effectiveness of the BIR model in contact recommendation, given that different methods may perform differently on different problems and domains. The adaptation of BIR to contact recommendation is:

$$f_u(v) = \sum_{w \in \Gamma(u) \cap \Gamma(v)} \log\left(\frac{|\mathcal{U}| - |\Gamma(w)| + 0.5}{|\Gamma(w)| + 0.5}\right) \tag{3.80}$$

where again, we map the triad (query, term, document) to the triad (target user, adjacent user, candidate user). In the previous equation, we may take the incoming links or the outgoing ones for the target and candidate users.

BM25

BM25 is one of the most effective probabilistic models in information retrieval up to date. This algorithm has its origin the Binary Independent Ranking model but, while the original algorithm considers that the frequency distribution of the terms in a document follows a Bernouilli distribution, BM25 considers that it follows a Poisson distribution (Sparck Jones et al., 2000). The similarity score between a document and the query by this method is the following:

$$sim(d,q) = \sum_{w \in d \cap q} \frac{(k+1) frec(w,d)}{k\left((1-b) + b\frac{len(d)}{avg_{d'}\, len(d')}\right) + frec(w,d)} RSJ'(w) \tag{3.81}$$

where $k > 0$ is a parameter that allows to tune the effect of the term frequency on the formula, $b \in [0,1]$ is a parameter that tunes the effect of the document length, and RSJ is the Robertson-Sparck Jones formula (see equation 3.80).

In contrast with the BIR model, BM25 takes into account both the length of the document and the frequency of its terms, giving place to a formula which is conceptually similar to the TF-IDF one.

The adaptation of this algorithm to user recommendation is similar to the previous cases: the new terms and documents are the network users, the frequency of a term in a document is the weight of the link between the "document user" and the "term user", and the length of the document is the sum of the weights of the edges between the user and all its neighbors. If we take uniform edge weights, the algorithm becomes:

$$f_u(v) = \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{(k+1)}{k\left(1 - b + b\frac{|\Gamma(v)|}{avg_{v'}(|\Gamma(v')|)}\right) + 1} RSJ(w) \tag{3.82}$$

Where $RSJ(w)$ is defined in the BIR score function. It is easy to see that the BIR model is equivalent to a particular case of this model, where $b = 0$.

Sometimes, better results are obtained for this algorithm as parameter k increases. Because of that, we shall also consider the limit of the above function where $k \to \infty$. We shall name this version Extreme BM25, and has the following definition:

$$f_u(v) = \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{RSJ'(w)}{1 - b + b\frac{|v|}{avg_{v'}(|v'|)}} \tag{3.83}$$

Statistical Language Models: Query Likelihood

A language model is a probability distribution over linguistic units, such as words, sentences or whole documents. Several IR models have been developed since the late 90' upon such distributions. In those models, a document is relevant to a query if the document model is likely to generate the query. One of the most widely used and successful approaches is the so-called query likelihood model (Ponte & Croft 1998).

$$f(d,q) = p(q|d) \propto \prod_{w \in \mathrm{q}} p(w|d) \propto \sum_{w \in \mathrm{q}} \log_2[p(w|d)] \tag{3.84}$$

Where it thus suffices to estimate the conditional probability $p(w|d)$ of a word given a document to compute the ranking function. This probability can be estimated by maximum likelihood with some smoothing to avoid the whole score of a document to vanish only because a single query word does not appear in the document.

A very common smoothing approach so-called Jelinek-Mercer is a parametrized mixture with the word prior over the whole collection (Jelinek & Mercer 1980)

$$p(w|d) = (1-\lambda)\frac{freq(w,d)}{len(d)} + \lambda\frac{\sum_{d' \in D} frec(w,d')}{\sum_{d' \in D} len(d')} \tag{3.85}$$

In our case, we adapt this algorithm in the same way as the previous ones, resulting in the following formulation:

$$f_u(v) = p(u|v) \propto \prod_{w \in \Gamma(u)} p(w|v) \propto \sum_{w \in \Gamma(u)} \log p(w|v) \tag{3.86}$$

where

$$p(w|v) = \frac{(1-\lambda)}{|v|} weight(w,v) + \lambda \frac{|\Gamma(w)|}{\sum_{x \in \mathcal{U}} |\Gamma(x)|} \tag{3.87}$$

And $weight(w,v)$ denotes the adjacency matrix:

$$weight(w,v) = \begin{cases} 1 & \text{if } v \in \Gamma(w) \\ 0 & \text{otherwise} \end{cases} \tag{3.88}$$

3.6 Content-Based Algorithms

We have thus far reviewed and proposed using only the information of the network structure to generate recommendations. There is often however further available information in social media that the recommender system can exploit, such as user-related contents and actions, e.g. tweets, retweets, hashtags, etc.

Even though prior studies have shown inferior performance of content-based methods compared to collaborative filtering for contact recommendation, we implement and include some content-based approaches in our experiments for comparison. In particular, we use a variation of the content-based approach proposed by Hannon et al. (2010) for their Twittomender system, which basically applies the IR vector-space model to represent tweets as vectors using the tf-idf term weighting scheme. That is, tweet vectors are built by equations (3.71) and (3.72). Where tweets play the part of text documents. Then, user vectors are built in the same vector space as centroids of a subset of tweets that are related to the user[4]:

$$\vec{u}_t = \sum_{tw \in tweets(u)} \text{tf-idf}(t, tw) \tag{3.89}$$

We consider four different possible definitions of the subset $tweets(u)$ to use in this method:

- The set of tweets published and retweeted by the user
- The set of tweets published and retweeted by the followers of the user
- The set of tweets published and retweeted by the followees of the user
- The set of tweets published and retweeted by both followers and followees of the user.

Finally, the scores of this algorithm are defined as the cosine similarity between the user centroids:

$$f_u(v) = \cos(\vec{u}, \vec{v}) = \frac{\sum_t \vec{u}_t \vec{v}_t}{\sqrt{\sum_t \vec{u}_t^2} \sqrt{\sum_t \vec{v}_t^2}} \tag{3.90}$$

[4] $\vec{u}_t$ can be indistinctly defined as a sum or an average, since the vector module is irrelevant for the cosine function which we shall use as the similarity function between users.

4. Evaluation metrics

The research and development of recommendation algorithms goes naturally hand in hand with the empirical evaluation of their effectiveness. Evaluation methodologies for recommender systems (and for that matter IR systems) have been at the centerpiece in the development of the field (Shani et al. 2015) for decades yet they remain an active and open area today, and a key concern in both research and commercial development, involving yet unsolved questions. Opening a new research direction in the recommendation field (as is contact recommendation) naturally – and unavoidingly – calls for specific work – and research oportunities – on evaluation in the scope of the new problem or problem variant.

An evaluation approach needs to, first of all, address the definition of what makes an algorithm better than another one.Traditionally, the ability of an algorithm to accurately predict the user's choices has been taken as the property that defines a good recommendation (Shani et al. 2015). While everyone agrees on accuracy being a required feature for any reasonable algorithm, it is nowadays not considered sufficient to provide useful recomnmendations: new complementary evaluation perspectives such as robustness, privacy, scalability, novelty and diversity have been increasingly put forward in recent years as fundamental additional dimensions to procure a complete enough assessment of the suitability of an algorithm.

In this chapter, we provide an overview of the different properties and dimensions one can consider in evaluating contact recommendation methods. We review traditional metrics as well as more recent ones, which have been developed in the IR and recommender systems fields. We furthermore propose novel adaptations of metrics from the field of social network analysis which provide new perspectives on the potential effects of recommendation algorithms in the structure and properties of the networks they act upon.

We study four different evaluation perspectives: the accuracy of the systems, the novelty and the diversity of the recommendations, and the effects of the recommendation over the properties of the network, the so-called structural diversity of the network (Huang et al. 2013).

4.1 Accuracy

The accuracy of the system is by far the most common evaluation perspective in the literature. A basic assumption in a recommender system is that a system that provides more accurate predictions will be preferred by the user (Shani et al. 2015).

The most common metrics are defined over the concept of relevance. Given a recommendation, we consider that an item is relevant if the user consumes that item. In contact recommendation, a recommended node is relevant if the target user creates a link to it. Several metrics are used for measuring the accuracy of a recommender system in terms of relevance.

Precision

Precision measures the fraction of retrieved items which is relevant to the target user (Baeza-Yates et al. 2011). It is computed as:

$$P(u) = \frac{|\text{relevant}(u) \cap \text{recommended}(u)|}{|\text{recommended}(u)|} = \\ = \frac{|\{v \in \mathcal{U} |\ v \in \mathcal{R}(u) \wedge (u,v) \in E_{test}\}|}{|\mathcal{R}(u)|} \tag{4.1}$$

Recall

This metric (Baeza-Yates et al. 2011) measures the proportion of the relevant items for the user which have been retrieved by a certain recomendation. This metric can achieve its maximum value, even when there are irrelevant items in the ranking. The formula for this metric is:

$$R(u) = \frac{|\text{relevant}(u) \cap \text{recommended}(u)|}{|\text{relevant}(u)|} = \\ = \frac{|\{v \in \mathcal{U} |\ v \in \mathcal{R}(u) \wedge (u,v) \in E_{test}\}|}{|\{v \in \mathcal{U} \mid (u,v) \in E_{test}\}|} \tag{4.2}$$

Normalized Discounted Cumulative Gain (nDCG)

This metric can be seen, in a way, as a refinement of precision. When a ranking is shown to an user, it is frequent for the last items on the list to go unnoticed. The lower the ranked position of a relevant recommendation, the less likely is that the target user will examine it. Additionally, for a single user, two recommendations can be relevant, but one can be more relevant than the other; that is, relevance (people's preference) is typically gradual rather than binary.

To model such considerations, Järvelin & Kekälainen (2000) proposed the nDCG metric. This metric rewards the occurrence of relevant contacts as higher as possible in the ranking, and considers different degrees of relevance for the items. It is defined as:

$$nDCG(u) = \frac{DCG(u)}{IDCG(u)} \tag{4.3}$$

with:

$$DCG(u) = \sum_{k=1}^{|R(u)|} \frac{g_u(v_k)}{\log_2(1+k)} \tag{4.4}$$

where v_k is the k-th user in the recommendation ranking, and $g_u(v_k)$ is the degree of relevance of the recommended user for the target user.

IDCG represents the value of the DCG metric for the best possible ranking for the items:

$$IDCG(u) = \max_{R \in \sigma(\mathcal{U})} DCG(u) \tag{4.5}$$

where $\sigma(\mathcal{U})$ represents all the possible orderings (permutations) of the set of candidate users.

4.2 Novelty

The novelty achieved by a system can be understood as the difference between what the system recommends and the past experience of a user (Castells et al. 2015). In the particular case of user recommendation, novelty means how different are the recommended users from the current user's friends and circles. For example, in Twitter, if a certain user follows many football players, and a Hollywood actress is recommended to him, that provides him more novelty than recommending further football players, or even other sports players.

Several metrics have been developed for measuring the novelty of recommender systems (Vargas et al. 2011). We adapt here the most representative ones for the user recommendation problem.

Popularity Complement (PC)

For a single user, this metric (Vargas et al. 2011) estimates the probability that the user did not know about the recommended users. It does so independently from the target user, by estimating a prior probability taking into account the observed connections between all users. The metric is defined as

$$PC = \frac{1}{|\mathcal{R}(u)|} \sum_{i \in \mathcal{R}(u)} \big(1 - p(known|i)\big) \tag{4.6}$$

$$p(known|i) = \frac{|U_i|}{|U|} = \frac{|\Gamma_{in}(i)|}{|U|} \tag{4.7}$$

In other words, in our context PC is inversely equivalent to the average in-degree of the recommended contacts.

Profile Distance (PD)

This metric measures the existing distance between the recommendations and the profiles of the users of the system (Vargas et al. 2011). The profile of a user is composed of the set of contacts the user has already interacted with (in this case, the adjacent nodes). This metric is computed using the following equation:

$$PD = \frac{1}{|U||\mathcal{R}|} \sum_{u \in U} \sum_{v \in \mathcal{R}(u)} d(u, v) \tag{4.8}$$

where $d(u, v)$ is the distance between the user v and the profile of user u. This distance is defined as the average distance between every neighbour in the user profile and v:

$$d(u, v) = \frac{1}{|\Gamma_{out}(u)|} \sum_{w:(u,w) \in E_{train}} dist(v, w) \tag{4.9}$$

This distance can be defined in any meaningful way for the domain at hand or the purposes of the application. For instance, it can be defined in terms of the common contacts, e.g. using the Jaccard similarity.

$$dist(v, w) = 1 - sim(v, w) \tag{4.10}$$

$$sim(v, w) = Jaccard(v, w) = \frac{|\Gamma_{out}(v) \cap \Gamma_{out}(w)|}{|\Gamma_{out}(v) \cup \Gamma_{out}(w)|} \tag{4.11}$$

In the context of user recommendation, this metric measures how overlapped are the distance two neighborhood of the target users, and the neighborhoods of the top candidate users. In case the value of this metric is high, this leads to the discovery of new neighborhoods, which may translate into new points of view for the target users.

4.3 Diversity

The diversity of a recommender system measures how different are the recommended items respect to each other (Castells et al. 2015). As an example, let's suppose there is a Twitter user who is a great fan of American basketball. Then, a recommendation which included players from different teams in the competition would be more diverse than another one which only recommended people from L.A. Lakers roster.

The evaluation of the diversity is a problem that has been addressed in both recommender systems (Vargas et al. 2011) and classical information retrieval ones (Clarke et al. 2008, Agrawal et al. 2009). We have adapted metrics from both sources to the problem we are dealing with.

Intra List Distance (ILD)

In the context of general recommender systems, Intra-list distance (ILD) measures the average distance between the different items in a recommendation ranking (Vargas et al 2011). In the specific case of contact recommendation, this metric can be understood as the average difference between the neighborhoods of the recommended users.

$$ILD(u) = \frac{1}{|\mathcal{R}(u)|(|\mathcal{R}(u)|-1)} \sum_{v\in\mathcal{R}(u), w\in\mathcal{R}(u), v\neq w} dist(v,w) \tag{4.12}$$

where $dist(v,w)$ is computed in the same way as the distance between users in PD. This metric is averaged over the full set of users:

$$ILD = \frac{1}{|U|}\sum_{u} ILD(u) \tag{4.13}$$

Gini coefficient

The Gini coefficient is commonly used to measure statistical dispersion in fields like ecology, economics or sociology. In recommender systems, this coefficient measures how evenly the items have been recommended to the different users (Vargas & Castells 2014). The Gini index is defined as follows:

$$G(s) = \frac{1}{|U|-1}\sum_{j=1}^{|U|}(2j-|U|-1)p(v_j|s) \tag{4.14}$$

where items are ordered by the number of times they have been recommended, and $p(v_j|s)$ is the probability of the j-th least recommended item being drawn from the recommendation lists generated by the recommender system s.

$$p(v \mid s) = \frac{|\{u|v\in\mathcal{R}(u)\}|}{\sum_v|\{u|v\in\mathcal{R}(u)\}|} = \frac{|\{u|v\in\mathcal{R}(u)\}|}{\sum_u|\mathcal{R}(u)|} \tag{4.15}$$

This metric is equal to 0 when all the items are equally recommended, and similar to 1 when a single item is recommended every time. To measure the diversity of the recommendation, we will take the value of this metric as

$$\text{Gini}(s) = 1 - G(s) \tag{4.16}$$

so the recommender will be more diverse if this coefficient is similar to 1, and less diverse if it is similar to 0.

4.3.1 Aspect-based diversity

In IR, the evaluation of the diversity of a system has been addressed from a different perspective, based on the different aspects (or subtopics) of queries or documents. This perspective has been recently proposed for the evaluation of recommender systems (Vargas et al. 2011, Castells et al. 2015), considering that, respectively, users and items play the role of queries and documents, i.e. considering the aspects of users and items (in our case, again, items and users do not differ).

The notion of aspect is abstract, and it can be particularized in many ways. For example, in the search field, aspects may be defined as query reformulations, ODP categories for the documents, Wikipedia disambiguations, subtopics defined by experts (like in TREC), etc. In our experiments, we explore user communities as a way to define the notion of aspect for the content recommendation problem. In other words, we consider that a recommendation is diverse if it includes people from different communities. In Chapter 5, we will explore how the community of a user is obtained.

From the multiple posibilities documented, we have considered two of the most representative metrics: subtopic recall and intent aware expected reciprocal rank (a more complex metric, deeply established in the field).

Subtopic Recall

Subtopic recall measures the proportion of subtopics which have been retrieved by the recommendation (Zhai et al. 2003). This measure is restricted to the recommended users which are relevant for the target user.

$$S - Recall(u) = \frac{1}{|\mathcal{Z}|}\left|\bigcup_{\substack{v \in \mathcal{R}(u) \\ (u,v) \in E_{test}}} \{z \in \mathcal{Z} | z \in aspects(v)\}\right| \tag{4.17}$$

In our context of use, subtopic recall computes the number of different communities which are recommended to the user.

Intent Aware Expected Reciprocal Rank (ERR-IA)

This metric is an *intent aware* version of the relevance metric known as Expected Reciprocal Rank (ERR, Chapelle et al. 2009). Intent aware metrics (Agrawal et al. 2009) adapt relevance metrics, such as ERR or nDCG to diversity metrics. For each aspect, intent aware metrics consider different probabilities that a certain user is interested on them. The metric is computed as the expected value of the relevance metric over the set of aspects:

$$\text{M-IA}(u) = \sum_{z \in \mathcal{Z}} p(z|u) M_z(u) \tag{4.18}$$

In the particular case of ERR-IA (Chapelle et al. 2011), we obtain a metric that, apart from considering different importance for the different aspects, penalizes both the occurrence of relevant documents for an aspect in higher ranked positions, and the repetition of the aspects in a rank. The metric is defined as follows:

$$\text{ERR-IA}(u) = \sum_{z \in \mathcal{Z}} p(z|u) ERR_z(u) \tag{4.19}$$

$$ERR_z(u) = \sum_{k=1}^{|\mathcal{R}|} p(rel|v_k, z) \prod_{j=1}^{k-1} \left(1 - p\left(rel|v_j, z\right)\right) \tag{4.20}$$

$$p(rel|v, z) = \frac{2^{g_z(v)} - 1}{2^{g_{max}}} \tag{4.21}$$

The probability that an intent is relevant to a user is defined as proportional to the presence of the intent in the items the user has in his profile (in our case, the probability that a community is relevant to a user is proportional to the number of neighbours that belong to that community).

$$p(z|u) = \frac{|\mathcal{I}_u \cap \mathcal{I}_z|}{\sum_{z'} |\mathcal{I}_u \cap \mathcal{I}_z|} \tag{4.22}$$

Where

$$\mathcal{I}_u = \{v \in \mathcal{U} \mid (u, v) \in E_{train} \vee (u, v) \in E_{test}\} \tag{4.23}$$

$$\mathcal{I}_z = \{v \in \mathcal{U} \mid z \in aspects(v)\} \tag{4.24}$$

4.4 Social Network Structure

Recommender systems over social networks have become a non-negligible force in the growth and evolution of the networks. For example, each month, the Who-To-Follow system in Twitter has been reported to lead to more than 500 million new connections (Goel et al. 2015). This fact makes motivates the analysis and evaluation of an additional dimension of the recommendation algorithms: their impact on the social network as a whole (Daly et al. 2010, Huang et al. 2013, Su et al. 2016).

Social network analysis and graph theory have provided a myriad of metrics to capture and understand the global structure and properties of networks (Newman 2010). In this work, we explore and analyze the adaptation of graph metrics to observe and measure the difference between recommendation algorithms in their impact on the network shape and relevant properties. Additionally, we have defined several new metrics seeking to capture specific angles in the effect of recommendation on the structure of the graph.

In most cases, the structural diversity metrics are defined over a network extended by the top k recommendations for each user, i.e. we consider that the user accepts those recommendations and creates edges to the recommended candidate users. We will assume that metrics are defined in that way by default. The rest of the metrics are only defined over the newly created edges, since defining them over the whole network only adds a constant to the final result.

Clustering coefficient

The clustering coefficient of a network measures the transitivity of the links of a network. It is computed as the ratio of transitive paths of lenght two over the total number of paths of lenght two. A path of length two is considered transitive if there is an edge from the origin of the path to the destiny.

$$CC = \frac{|\{(u,v,w)|u,v,w \in \mathcal{U} \wedge (u,v),(v,w),(u,w) \in E\}|}{|\{(u,v,w)|u,v,w \in \mathcal{U} \wedge (u,v),(v,w) \in E\}|} \tag{4.25}$$

This metric is related to the redundancy of the edges, so we consider that the network is more diverse when this coefficient has a low value.

It is also possible to define a clustering coefficient for a single vertex, u. This clustering coefficient is defined as the probability that if we randomly pick an incident node to u, v, and an adjacent node, w, user v has a link to w. This is known as the local clustering coefficient. This can be averaged by user, providing an alternative formulation of the global clustering coefficient (Watts & Strogatz 1998):

$$ALCC(u) = \frac{1}{|\mathcal{U}|}\sum_{u \in \mathcal{U}} \frac{|\{(v,w)|\ v \in \Gamma_{in}(u) \wedge w \in \Gamma_{out}(u) \wedge (v,w) \in E\}|}{|\Gamma_{in}(u)||\Gamma_{out}(u)| - |\Gamma_{in}(u) \cap \Gamma_{out}(u)|} \tag{4.26}$$

Compared to the previous definition of the global clustering coefficient, ALCC (for average local clustering coefficient) tends to be dominated by vertices with low degree, since the denominator of the clustering coefficient is small. As a consequence, although both definitions tend to correlate in general, they may diverge to some extent sometimes.

Edge embeddedness

The embededness of an edge measures how redundant is that edge, and is computed as the Jaccard coefficient of the neigborhoods of the edges

$$\begin{aligned} \text{Embeddedness}\big((u,v)\big) &= \text{Jaccard}(\Gamma_{out}(u) - \{v\}, \Gamma_{in}(v) - \{u\}) \\ &= \frac{|(\Gamma_{out}(u) - \{v\}) \cap (\Gamma_{in}(v) - \{u\})|}{|(\Gamma_{out}(u) - \{v\}) \cup (\Gamma_{in}(v) - \{u\})|} \end{aligned} \tag{4.27}$$

We also measure the number of edges with embeddedness equal to zero (edges known by Granovetter (1973) as the local bridges of the network).

$$\text{LocalBridges}(G) = \big|\big\{(u,v) \in E|\ \text{Embeddedness}\big((u,v)\big) = 0\big\}\big| \tag{4.28}$$

Average Shortest Path Length (ASL)

Average shortest path length measures how close is a user from the rest of the network. The distance between two users might be infinite (you cannot reach the second node from the first). This is computed as the average over the finite paths (Newman 2010).

$$ASL(G) = \frac{1}{|\{(u,v)|u \in \mathcal{U}, v \in \mathcal{U}, d(u,v) < \infty\}|}\sum_{u \in \mathcal{U}}\sum_{v \in \mathcal{U}} \tilde{d}(u,v) \tag{4.29}$$

$$\tilde{d}(u,v) = \begin{cases} d(u,v) & \text{if } d(u,v) < \infty \\ 0 & \text{if not} \end{cases} \tag{4.30}$$

Eccentricity

The eccentricity of a node u is defined as the maximum distance between that node and any other node in the network (Danklemann et al. 2012):

$$\text{ecc}(u) = \max\{d(u,v)|d(u,v) < \infty\} \tag{4.31}$$

This individual metric represents the maximum cost for a message to go from a user from any other node in the network. Depending on how we aggregate this metric over the complete network, it is possible to define several metrics:

- **Diameter:** It is defined as the maximum value for the eccentricity of the graph

$$\text{diameter}(G) = \max_{u \in \mathcal{U}}\big(ecc(u)\big) = \max(\{d(u,v) | d(u,v) < \infty\}) \tag{4.32}$$

- **Average eccentricity:** As its name indicates, it is computed as the mean eccentricity over the network.

$$\text{avg-ecc}(G) = \frac{1}{|\mathcal{U}|}\sum_{u \in \mathcal{U}} \max(\{d(u,v) | d(u,v) < \infty\}) \tag{4.33}$$

Recommendation distance

This metric measures the average distance between the target users and the recommended ones in the train set.

$$d_{rec}(G) = \frac{1}{\left|\tilde{E}_{rec}\right|}\sum_{(u,v) \in \tilde{E}_{rec}} d_{train}(u,v) \tag{4.34}$$

where

$$\tilde{E}_{rec} = \bigcup_{u \in \mathcal{U}} \{v \in \mathcal{R}(u) |\, d_{train}(u,v) < \infty\} \tag{4.35}$$

The number of new edges recommended at infinite distance is also computed:

$$edges_{\infty}(G) = |E_{rec}| - \left|\tilde{E}_{rec}\right| = \left(\sum_{u \in U} |\mathcal{R}(u)|\right) - \left|\tilde{E}_{rec}\right| \tag{4.36}$$

Degree assortativity

The degree assortativity measures to what extent nodes with similar degree join themselves (Newman 2010, Ch. 7). It is computed as the Pearson correlation of the degrees of the nodes. We consider two variants of this metric:

- **Undirected assortativity:** This variant considers every edge as if the graph was not directed.

$$\text{u-assort}(G) = \frac{2m\sum_{i \to j}|\Gamma(i)||\Gamma(j)| - \left(\sum_j |\Gamma(j)|^2\right)^2}{4m\ \sum_i |\Gamma(i)|^3 - \left(\sum_j |\Gamma(j)|^2\right)^2} \tag{4.37}$$

- **Directed assortativity:** This metric computes the assortativity of the graph considering the direction of the edges. Its general formula is the following:

$$\text{d-assort}(G) = \frac{\frac{1}{m}\sum_{ij}\left[A_{ij} - \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m}\right]x_i x_j}{\sigma_{out}\sigma_{in}} \tag{4.38}$$

where x_i, x_j can take as values $|\Gamma_{out}(i)|, |\Gamma_{in}(i)|$ and

$$\sigma_{in}^2 = \sum_{ij}\left(|\Gamma_{in}(j)|\delta_{ij} - \frac{|\Gamma_{in}(i)||\Gamma_{in}(j)|}{m}\right)x_i x_j \tag{4.39}$$

$$\sigma_{out}^2 = \sum_{ij}\left(|\Gamma_{out}(j)|\delta_{ij} - \frac{|\Gamma_{out}(i)||\Gamma_{out}(j)|}{m}\right)x_i x_j \tag{4.40}$$

In our experiments, we shall test the in-degree assortativity of the network (taking x_i as $|\Gamma_{in}(i)|$ and x_j as $|\Gamma_{in}(j)|$), since the out-degree is equally increased for all the users in the network.

4.4.1 Weak ties

The notion of weak tie and its possible value in terms of social welfare has been one of the problems which has attracted more attention from scholars and people interested in social network analysis or sociology since the early 1970s (Granovetter 1973, Aral 2016). Since Granovetter (1973) hypothesized that weak links in network provide more novel information than strong links, recommending them may improve the social welfare of the users in the network, in terms of novelty and diversity of the information that arrives to the different users.

This hypothesis encourages the analysis of the role of weak ties in contact recommendation, which is one of the main focus of the present work. Two different perspectives are considered for this analysis: the first one, which will be explored in this chapter and the next, seeks to explain the number and distribution of weak links in the recommendations. The other perspective, which will be explored in chapter 6, explores the effects of recommending weak ties in the information diffusion over the networks.

Before performing that analysis, we have to decide which definition to use for the concept of weak tie. Since the different definitions by Granovetter are very restrictive, we will use the alternative definition proposed by De Meo et al. (2014), who defined a weak tie as an edge between two different communities in the network. Under that definition, the recommendation of weak ties is related to the concept of novelty of the recommendation: links which travel between two different communities escape the close environment of the users on each endpoint of the link, so it is less likely for the user to know the recommended user. In fact, as we will see in the next chapter, the number of weak links in the network is highly correlated to the novelty metric known as profile distance (PD).

Using the previous definition of weak tie, we define several metrics:

Weak ties

This metric is the simplest one related to weak ties. It simply measures the number of new links between communities created by the recommendation. We only defined this metric over the newly created links, since all the weak links in the training graph still remain after the recommendation.

$$WT = |\{(u,v) \in E \mid u \in \mathcal{U}, v \in R(u) \wedge comm(u) \neq comm(v)\}| \tag{4.41}$$

Modularity

Given a division of the social network in several communities, this metric measures the quality of that division (Newman et al. 2004). This measure is related to the concept of homophily (users tend to group themselves with similar users).

The modularity is measured as the number of edges inside communities (strong links), in comparison with the expected number of strong ties we would find if the edges were placed at random. The metric is normalized by the maximum value for the metric (considering that all edges are between nodes in the same community).

$$\mathrm{mod}(G) = \frac{\sum_{ij}\left(A_{ij} - \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m}\right)\delta(c_i, c_j)}{m - \sum_{i,j}\frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m}\delta(c_i, c_j)} \tag{4.42}$$

Community Gini

Both weak ties and modularity provide a measure about how many links exist between communities. However, these metrics do not provide information about how these weak ties are distributed over the network. As an example, Figure 12 contains two graphs with the same 3 communities (represented in green, blue and brown), and the same value for the modularity and the number of weak ties (red edges in the figures). However, the distribution of those weak ties is very different. In the above graph, the three weak ties go from the blue community to the green community, while the brown one is always disconnected. In the other one, each tie connects two previously unconnected communities.

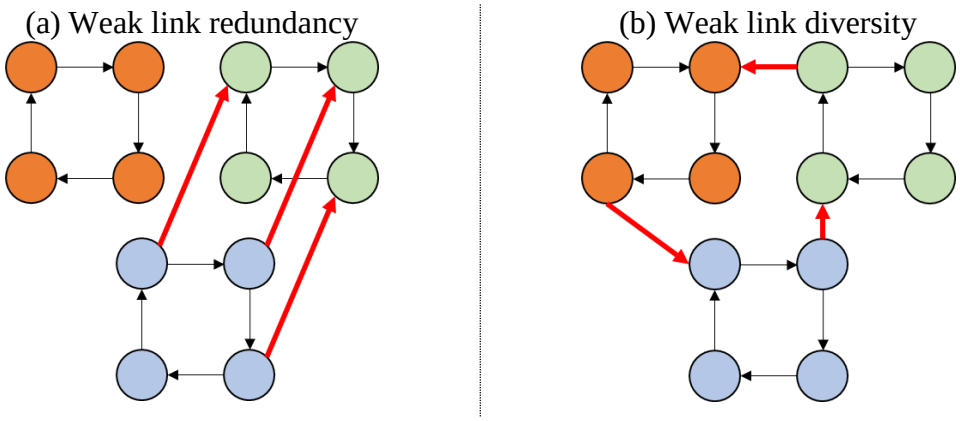


Figure 12. Graphs with the same modularity

A network with its weak ties distributed evenly between the different communities (as the second one in the picture) seems more diverse than another one with all the edges between only two communities (as the first one). This metric offers a global vision of the distribution of the weak links, and acts as a complement for the modularity (in a diverse network, we would want low modularity and high community Gini). Depending on which endpoints of the links we observe, we distinguish between two variants of this metric.

- **Community In-Gini:** This variant is similar to the Gini coefficient metric defined for diversity, if we considered each community as a different user, and each link between communities as a recommendation, i.e. this metric measures how evenly are recommended the communities to users in different ones.

$$CIGini(s) = 1 - \frac{1}{|\mathcal{C}| - 1}\sum_{j=1}^{|\mathcal{C}|}(2j - |\mathcal{C}| - 1)p(c_j|s) \tag{4.43}$$

$$p(c \mid s) = \frac{|\{(u,v) \in E | comm(u) \neq c \wedge comm(v) = c\}|}{\mathrm{WT}} \tag{4.44}$$

- **Community Pair-Gini:** This metric represents an extension of the previous measure, which, instead of just observing the destination communities of the weak links, takes each one of the links as a pair of communities (c_1, c_2) with $c_1 \neq c_2$, and studies how are those pairs distributed.

$$CPGini(s) = 1 - \frac{1}{|\mathcal{C}|(|\mathcal{C}|-1)-1} \sum_{j=1}^{|\mathcal{C}|(|\mathcal{C}|-1)} (2j - |\mathcal{C}|(|\mathcal{C}|-1) - 1) p\big((c_1, c_2)_j | s\big) \tag{4.45}$$

where $\mathcal{C}$ is the set of communities, $c_1 \neq c_2$ and

$$p\big((c_1, c_2) \mid s\big) = \frac{|\{(u,v) \in E | comm(u) = c_1 \wedge comm(v) = c_2\}|}{\mathrm{WT}} \tag{4.46}$$

5. Experiments

One of the main goals of our investigation consists in providing a thorough empirical comparison between a comprehensive set of representative state of the art contact recommendation approaches, plus the original methods we have proposed in previous chapters. We have conducted in terms of the most classical evaluation perspective, the accuracy, but also in terms of new perspectives, along novelty and diversity dimensions of recommendation. In this chapter, we describe the experimental design we develop for the empirical study, and we report the obtained results along with an extensive analysis thereof.

5.1 Data Sets

Our empirical work focuses on data from the Twitter[5] social network. Twitter is a microblogging platform, launched in 2006, which enables users to post messages (known as **tweets**) of up to 140 characters. These messages can contain pictures, URLs or videos, and can be categorized using tags. All of these tags start with the character # and they are known as **hashtags**. An example can be seen in Figure 13.

Figure 13. A tweet with a hashtag

Twitter users and their connections form an asymmetric social network. By creating an edge (u, v) in the network, the contents published by the of the second user, v, (known as **followee**) are shown in the **timeline** of user u (known as **follower**). The timeline of a user is just a collection of the tweets published by his followees, chronologically ordered newest to oldest.

Users can also interact between them using the platform capabilities. Most of those interactions are made through the use of tweets. There are three types of these interactions: **retweets**, **mentions** and **replies**. A retweet is the way for forwarding content from other users. The tweet is published in the user's profile, while the authorship of the tweet is still assigned to the original creator. An example of a retweet is shown in Figure 14. A mention, as its name indicates, consists on including a reference to another Twitter user in the content of the tweet. It is done by writing "@" before the nickname of that user (see Figure 15). Finally, replies allow users to answer other people's tweets. An example of a reply is shown in Figure 16.

An implicit graph can be obtained from those interactions. This **interaction graph** contains the same users as the explicit follows graph, and a user A contains a link to another user, B, if A has interacted with B (if A has retweeted, mentioned or replied B).

[5] Twitter: https://twitter.com

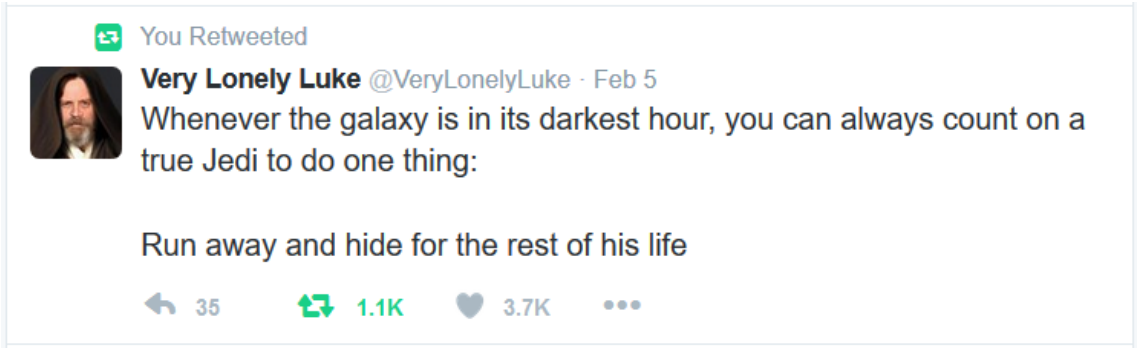


Figure 14. Retweet example

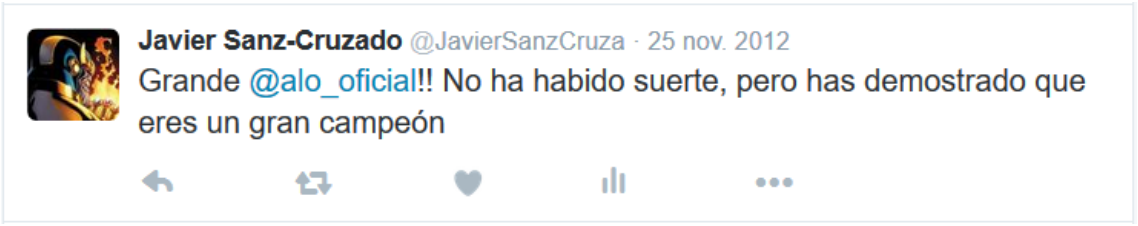


Figure 15. Mention example

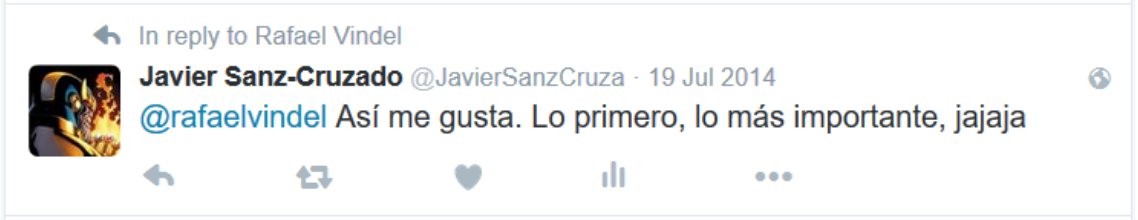


Figure 16. Reply example

The huge dimensions of the complete network (175 million active users and approximately twenty billion edges in 2012, according to Myers et al. 2014), make working with the complete network unfeasible for most people. Also, at least to date, only Twitter has access to the full network of users. Researchers and most business therefore generally work on reduced samples of the network, computationally manageable. This is also our case here. For our research, we have collected two different samples of the Twitter network. In this section, we provide information about how this data was collected, as along with a full description of the different datasets.

5.1.1 Preparation of the Data Sets

First, we describe how the data was obtained and prepared for the different experiments we have carried out. In particular, we document the crawling from Twitter, the data partitioning and cleaning, and the indexing of the retrieved tweets.

Mining Twitter data

Twitter data samples can be obtained through one of the official APIs provided by Twitter. The simplest and most widely-known API is the REST API[6]. This API allows its users to crawl most parts of the information contained in the Twitter network (tweets, user information, follows relations, interactions contained in tweets, hashtags, etc.). However, it has several limitations: A set of restrictions is related to the API usage.

[6] Twitter REST API: https://dev.twitter.com/rest/public

Depending on the function, each 15 minutes, Twitter API can only be successfully called from 15 to 900 times. These limitations make important to retrieve the greatest amount of information each time we make a call to the API. Then, if someone aims to retrieve all the tweets posted by a set of users in a certain period of time, this might not be possible: for each user, only the last 3200 tweets published by a certain user can be retrieved. This is not especially problematic for most users, but there are some accounts (mostly a subset of the so-called verified accounts) like sports teams' ones, which publish way faster than average, quickly passing that limit, thus limiting the information which can be retrieved about them. Also, the tweets of the users can only be retrieved if their account is public (everyone can see the user's tweets). Both facts make impossible to retrieve the complete set of interactions of every user in the network.

Despite its limitations, the Twitter API provides access to a wealth of real social network data at unprecedent scale richness and openness, which thousands of researchers and practitioners worldwide are extensively taking advantage of (Krishnamurthy et al. 2008, Kwak et al. 2010).

In our experiments, the main data download was aimed at retrieving the network of interactions of the network. There are many possible methods for sampling social networks (Leskovec et al. 2006, Das et al. 2008). In our case, we have implemented a variant of the method known as snowball sampling (Goodman 1961). This method starts the sampling by selecting a seed user (or a set of them). Then, for each user in the seed set, the method selects at most k different neighborhoods. This set of newly retrieved users form the first wave of the sample. Then, the process is repeated with the users in the first wave, and so on, until the sample contains the desired number of users.

Our method differs from pure snowball sampling in the selection of the neighbors. Instead of asking for a fixed number of neighbors, we expand a single user by taking every interaction we have observed in a certain collection of tweets that the corresponding user has hosted. We consider two different criteria for selecting those sets: a) selecting a fixed number of the most recent tweets that a certain user has published (for example, the maximum number of tweets which can be retrieved using a single API call), and b) taking all the tweets in a given time interval. Apart from those tweets, all the retweeted tweets and the replied ones are also retrieved, even when they are not included in the time interval or the given number of tweets. Along with all the tweets, users and interactions, the crawler also retrieves the set of hashtags and URLs contained in the tweets.

An example of this algorithm is illustrated in Figure 17a. The green node represents the seed user of the sampling, light blue nodes represent the first wave, dark blue the second, black the third and red represents unreachable nodes (the seed user has interacted with them, but the sampling of tweets has not discovered the interaction). Edges to previously discovered users are shown in black, edges to new nodes are shown in blue, and edges to undiscovered nodes are shown in red.

When the sampling has found the desired number of nodes, the algorithm stops. Then, several nodes Then, all the links between nodes (in the given number of tweets or time interval) in the sample are retrieved. This provides a closure for the graph. An example is shown in Figure 17b, where the closure links are shown in orange.

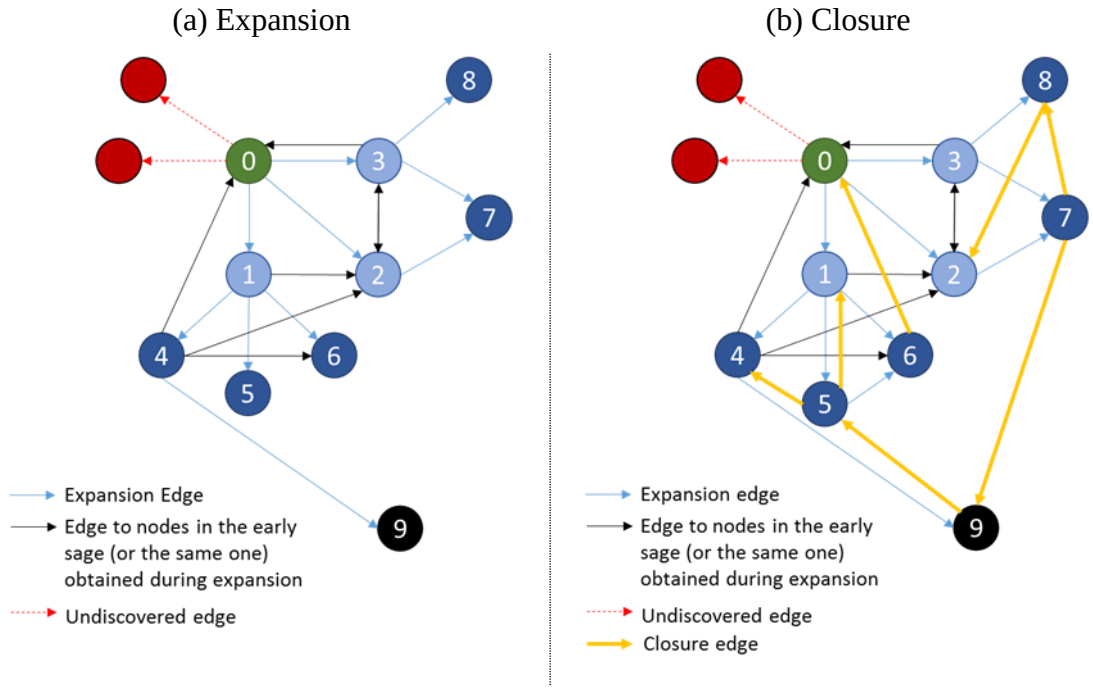


Figure 17. Sampling example

After the interactions graph is retrieved, we can download the follows relationships between the collected users, by querying the Twitter API for the presence of such links between each pair of nodes.

Graph partitioning and cleaning

Once the data has been retrieved from the Twitter API, it is necessary to prepare it for the experiments. For that, we first assign weights to the different edges in the network. There are many possibilities for these weights, like the number of interactions between those users, the redundancy of the edges, etc. To simplify, we have considered that every existing link in the network has weight equal to 1.

$$weight(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

Since all our experiments are offline tests, it is necessary to obtain two different subsets of links: the training and test sets. For obtaining them, we perform a temporal split of the graph where we take only those nodes with at least one link (incoming or outgoing). We differentiate how we split the graphs depending on their type (follows or interactions).

As we explained in the previous section of this work, the interactions between users are directly retrieved from the tweets published by the user. Twitter establishes a temporal mark in everyone of those tweets, so it is easy to create a temporal partition. First, we select a certain date. Every interaction which was found before that date goes to the training set, and every interaction after that date is pushed to the test set.

In the case of the follows graph, Twitter does not provide temporal information about when a user started following another one, which makes imposible making a temporal partition only with a download of data. As we want to make a temporal partition, two downloads of the follow graph are made in different time, using the previously explained procedure. The links in the first download form the training graph, and the newly created edges in the second download form the test set.

Finally, once both training and test edges are separated, it is necessary to clean the test graph to prevent errors. We remove three different sets of links:

- **Links appearing on both training and test sets:** The target users already know about these nodes (the user has interacted with them, or has followed them), so it makes no sense recommending them.
- **Links containing nodes that are not present on the training set:** These links cannot be guessed by the different recommendation algorithms, since they do not have information about them.
- **Reciprocal edges to the ones in the training set:** We detected a bias to those links on the different recommendation algorithms which blurred the effects of the algorithms, so we decided to remove those edges, and recommend only not reciprocal edges.

Indexes

One of our recommendation approaches, the one known as Hannon, uses the content of the different tweets of the users in the network. To simplify the access to that content, we generate an index for each dataset using Apache Lucene, a Java library for building indexes and search engines. Every retrieved tweet was included in that index, treated as a different document.

5.1.2 Attribute Spaces

As we indicated in section 4.3, the evaluation of aspect-based diversity metrics like α-nDCG or ERR-IA requires the definition of attribute spaces for the different users. AS the attributes for the users, we use the different communities detected in the training graph by a set of specialized algorithms. We have selected three of the most succesful, fast and well-known community detection algorithms in the literature: Leading Eigenvector (Newman 2006), Louvain (Blondel et al. 2008) and Infomap (Rosvall et al. 2008).

Next, we show how these methods find a good partition for the graph:

Louvain

Louvain method (Blondel et al. 2008) is a community detection method oriented to large graphs. The algorithm starts assigning each node in the network a different community. Then, two phases are iteratively repeated for optimizing the modularity. An example that shows a couple of iterations can be observed in Figure 18.

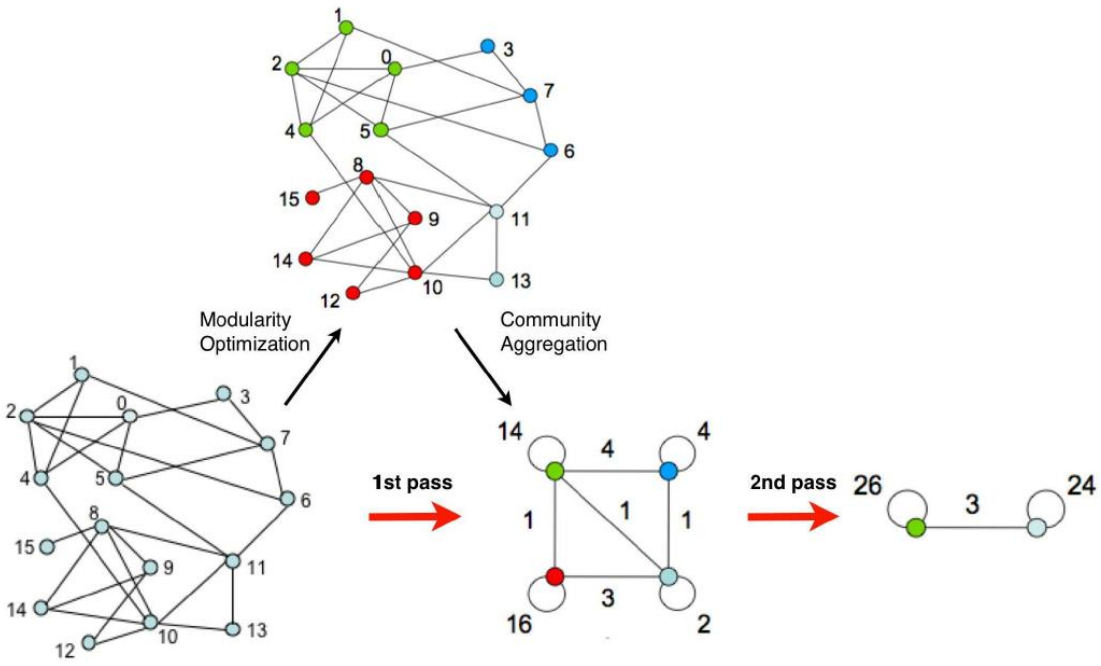


Figure 18. Louvain community detection algorithm (from Blondel et al. 2008)

In the first phase, for every node i in the network, the modularity gain resulting from changing the community of this node to each one of its neighbours is computed. Then, the community of node i is changed to the community of node j where it achieved the maximum value for the gain (if this value is positive. Otherwise, the community stays the same). This is repeated until the modularity does not increase.

The second phase repeats the same algorithm over a new graph. The nodes of this new graph are the communities of the original graph, and the edges are weighted as the sum of the weights of the links between communities.

In both phases, the modularity gain is computed as

$$\Delta(mod(G)) = \left[\frac{\Sigma_{\text{in}} + k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)\right] + \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right] \tag{5.2}$$

where Σ_{in} represents the sum of the link weights inside the community c, $k_{i,in}$ is the sum of the weights of the nodes from node i to nodes in the community c, Σ_{tot} is the sum of the weights of the incident links to the community, k_i is the sum of the weights of the incident links to node i, and m is the sum of the weights of all links in the community.

Infomap

Infomap (Rosvall et al. 2008) relates the community detection problem with the minimization of the length of a code used for describing a random walk in the network. This code uses a two-level Huffman compressing code (Huffman 1952): the first level differentiates the different communities in the network, and the second one differentiates the nodes inside each community. The idea for minimizing the code is the following: if there are few links between communities, a random walker is more likely

to stay inside of each community, so random walks may be described only by the second level code, leading to a more compact representation.

Given a network partition in communities, $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$, the average length of the description of a step in an infinite random walk (the function to minimize) is defined as:

$$L(\mathcal{C}) = q_{\curvearrowright} H(Q) + \sum_{i=1}^{m} p_{\circlearrowleft}^{i} H\left(P^i\right) \tag{5.3}$$

Where $q_{\curvearrowright}$ represents the probability of exiting a node in each step, $H(Q)$, the entropy of the module names, represents a lower bound for the size of the code used for naming a community, $p_{\circlearrowleft}^{i}$ is the fraction of the movements inside a community which occur in community c_i, and $H\left(P^i\right)$, the entropy of the movements inside the community c_i represent a lower bound for the average length of the codeword for the name of a node in that community.

Leading Vector

Leading Vector (Newman 2006) reformulates the modularity of a graph in terms of matrices, which changes the formulation of the optimization problem to a spectral problem in linear algebra. This algorithm recursively divides a graph in two different communities, C_1 and C_2. The modularity of the graph is computed in terms of a special matrix, B, called the modularity matrix of the graph. This matrix is defined as:

$$B_{ij} = A_{ij} - \frac{|\Gamma(u_i)|\left|\Gamma\left(u_j\right)\right|}{2|E|} \tag{5.4}$$

where A represents the adjacency matrix of the graph.

The modularity of the graph is reformulated in terms of the eigenvalues β_i and the eigenvectors b_i of the modularity matrix as:

$$mod(G) = \frac{1}{4m} \sum_{i} (b_i^T s)^2 \beta_i \tag{5.5}$$

where s is a vector whose i-th coordinate is equal to 1 if the i-th node belongs to the first community, and -1 if it belongs to the second one. In order to optimize the modularity, the coordinates for vector s have to be selected. The choice that maximizes the modularity consists on taking that vector equal to the sign of the coordinates of the leading eigenvector (the eigenvector of the largest eigenvalue).

The recursion is continued until the contribution of the partition to the modularity is negative. In that case, that part of the graph is not subdivided.

5.1.3 Description of the Data Sets

Using the sampling method described above, we have selected two different datasets. Both datasets differ on the selection of tweets: the first one retrieves every tweet the explored user has published during a certain month, and the second one retrieves the last 200 tweets from the user to expand (the maximum you can obtain from a unique Twitter API call). In the rest of the document, the first dataset is known as 1 Month and the second one as 200 Tweets.

1 Month

The first dataset contains every tweet published by the retrieved users between 16th June 2015 and 16th July 2015. It was retrieved starting from user @j_yubero, and the set contains a total of 10,019 users and 2,369,596 tweets. From the full set of tweets, only 550971 contain one or more of the 107,332 unique hashtags retrieved. 741,679 interactions were detected (between retweets, replies and mentions). Those interactions are transformed into 234,869 edges in the interaction graph (shown in Figure 19). The follow graph (Figure 20) which corresponds to the users in this dataset was retrieved in 9th October 2015 (training set) and 9th February 2016 (test set) and it contains a total of 633,387 links between nodes.

As the dataset is mainly formed by four weeks of tweets (excepting the older tweets, retrieved via retweets or replies), the selected date for the interactions graph split was the end of the third one (9th July 2015 at 19:29:59). The first three weeks were selected to create the training set, and the last one forms the test set. Table 1 shows the number of unique users and edges on each one of the sets. It must be noticed that, from the 7,902 users in the test set, only 7,114 of them have created new links.

	Nodes	**Edges**
Train	9,528	170,425
Test	7,902	57,846

Table 1. Interactions 1 Month partition

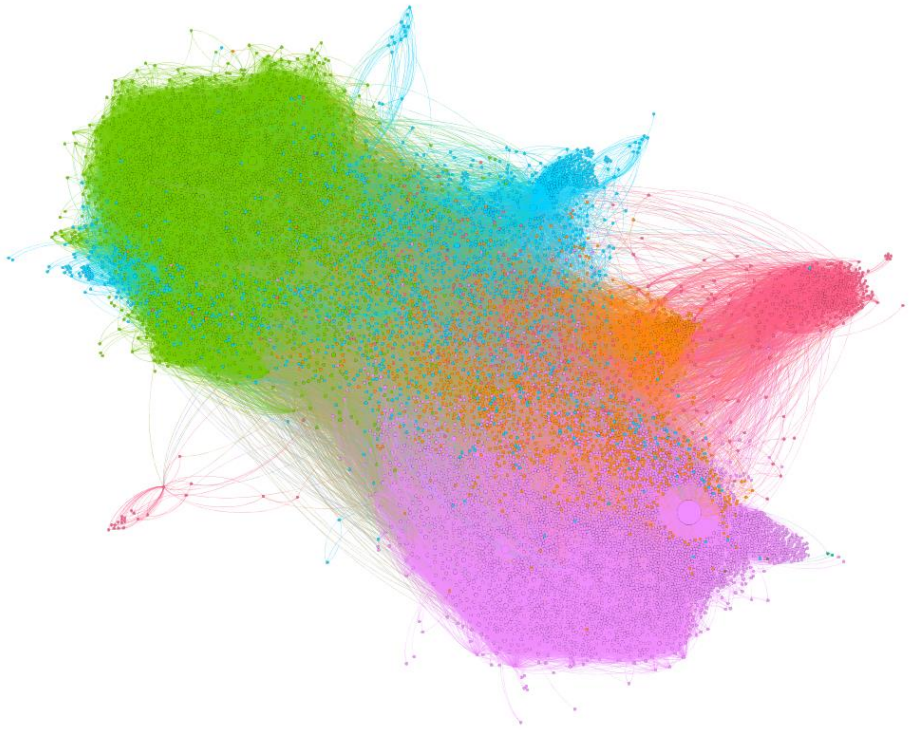


Figure 19. Complete 1 Month interactions graph (Colors represent Louvain communities detected on the training graph).

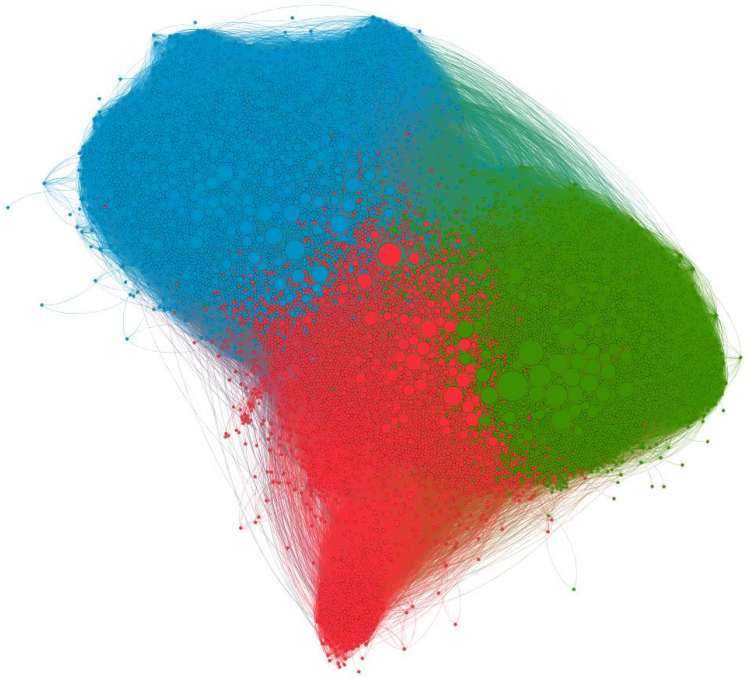


Figure 20. 1 Month follows graph (Colors represent Louvain communities detected on the training graph)

Table 2 shows the number of users and links in the training and test sets for the follows graph. The partition was made, as explained before, by taking as test set the links from the second download that were not in the first one. In this partition, the difference between the number of edges in both sets is notably higher than in the interactions graph partition. 6,054 nodes from the test set have outgoing links in that set.

	Nodes	**Edges**
Train	9,861	645,000
Test	6,419	13,766

Table 2. 1 Month follows partition

Table 3 show a brief description of the training graphs in terms of social network analysis metrics. It can be observed that the average degree (and therefore the density of the graph) is higher in the follows graph than in the interactions graph. This leads to a slightly shorter paths for that graphs. The redundancy and transitivity of the networks are also higher in the follows graph.

The degree distributions of both training graphs can be observed in Figure 26. Both the in-degree and the out-degree distributions of the interactions graph are more skewed than the distributions for the follows graph.

In Table 4, the different metrics about communities for the interactions graph are shown. It can be noticed that modularity and the number of weak ties between communities are very similar for the three community detection algorithms (it differs in the case of the strongly connected components). The main difference between the algorithms comes from the number of detected communities and the Gini metrics.

Metric	Interactions	Follows
Density	0.0018	0.0065
Average Degree	17.89	65.4092
Clustering Coefficient	0.06	0.22
Average Local Clustering Coefficient	0.18	0.25
Edge Embeddedness	0.0242	0.0512
Average Shortest Path Length	3.54	3.135
Diameter	13	11
Average Eccentricity	6.67	6.23
Strongly Connected Components	1,702	1,715
Biggest SCC	81.8%	82.4%

Table 3. Training set metrics (1 Month)

Algorithm	Num.	Modularity	Weak Ties	In-Gini	Pair-Gini
Louvain	8	0.7073	32,847	0.4194	0.1698
Leading Vector	3	0.7485	25,135	0.9245	0.7871
Infomap	135	0.7314	29,679	0.1010	0.0112
SCC	1,702	0.0251	18,007	0.0408	0.0002

Table 4. Community metrics (1 Month interactions)

Figure 21 and Figure 22 display the percentage of nodes in each one of the communities detected by Louvain and Leading Vector. Instead of a pie chart, Figure 23 shows the distribution of the Infomap communities depending on the size of the community. We observe that Infomap has the greatest number of communities, but most of them are very small. Louvain and Leading Vector have a smaller number of communities, but the number of nodes of each one is more equally distributed.

Table 5 shows the community-related metrics for the follows graph. In this graph, we observe greater differences between Louvain, Infomap and Leading Vector in the number of weak ties. The rest of the metrics behave as they did for the interactions graph. For the three algorithms, the number of communities is smaller than it was in the interactions graph. The sizes of each community can be observed in Figure 24, Figure 25 and Figure 27.

Algorithm	Num.	Modularity	Weak Ties	In-Gini	Pair-Gini
Blondel	4	0.6509	143,333	0.4900	0.3262
Newman	2	0.7385	82,316	0.8282	0.8282
Infomap	49	0.6756	133,238	0.0724	0.0109
SCC	1,715	0.0147	45,703	0.0043	0.0003

Table 5. Community metrics (1 Month follows)

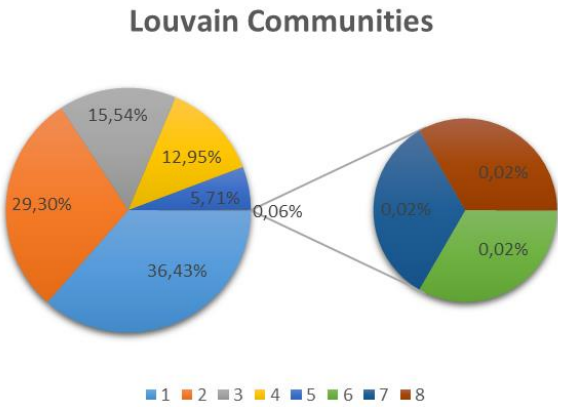


Figure 21. Louvain community sizes (1 Month interactions)

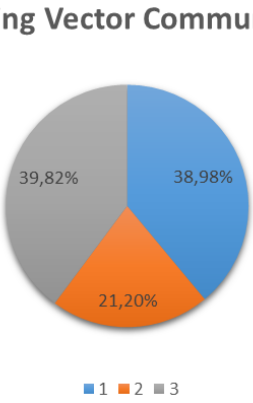


Figure 22. Leading Vector community sizes (1 Month interactions)

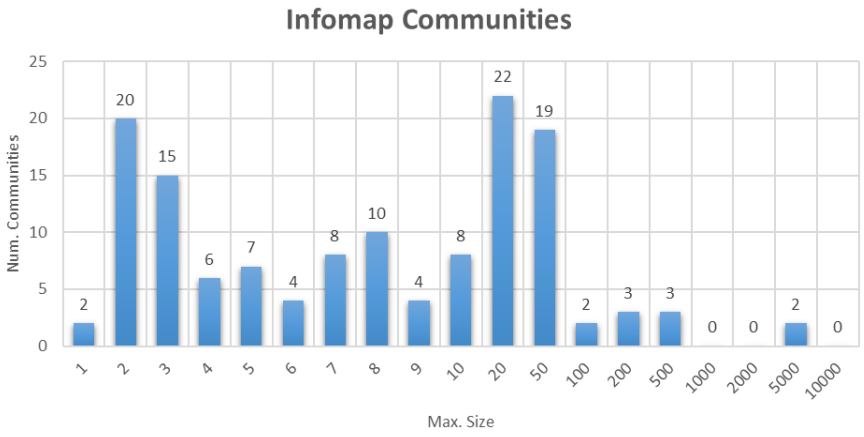


Figure 23. Infomap community distribution (1 Month interactions)

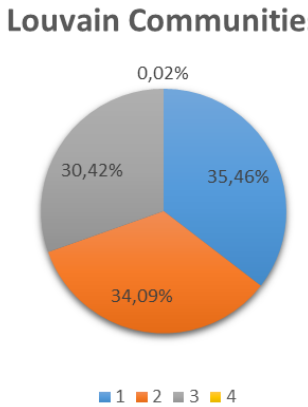


Figure 24. Louvain community sizes (1 Month follows)

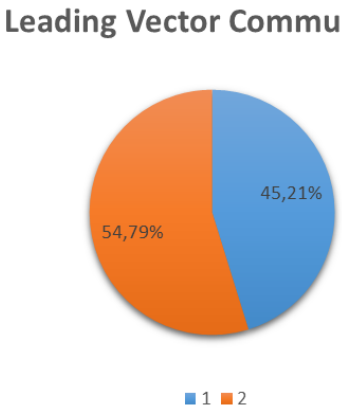


Figure 25. Leading Vector community sizes (1 Month follows)

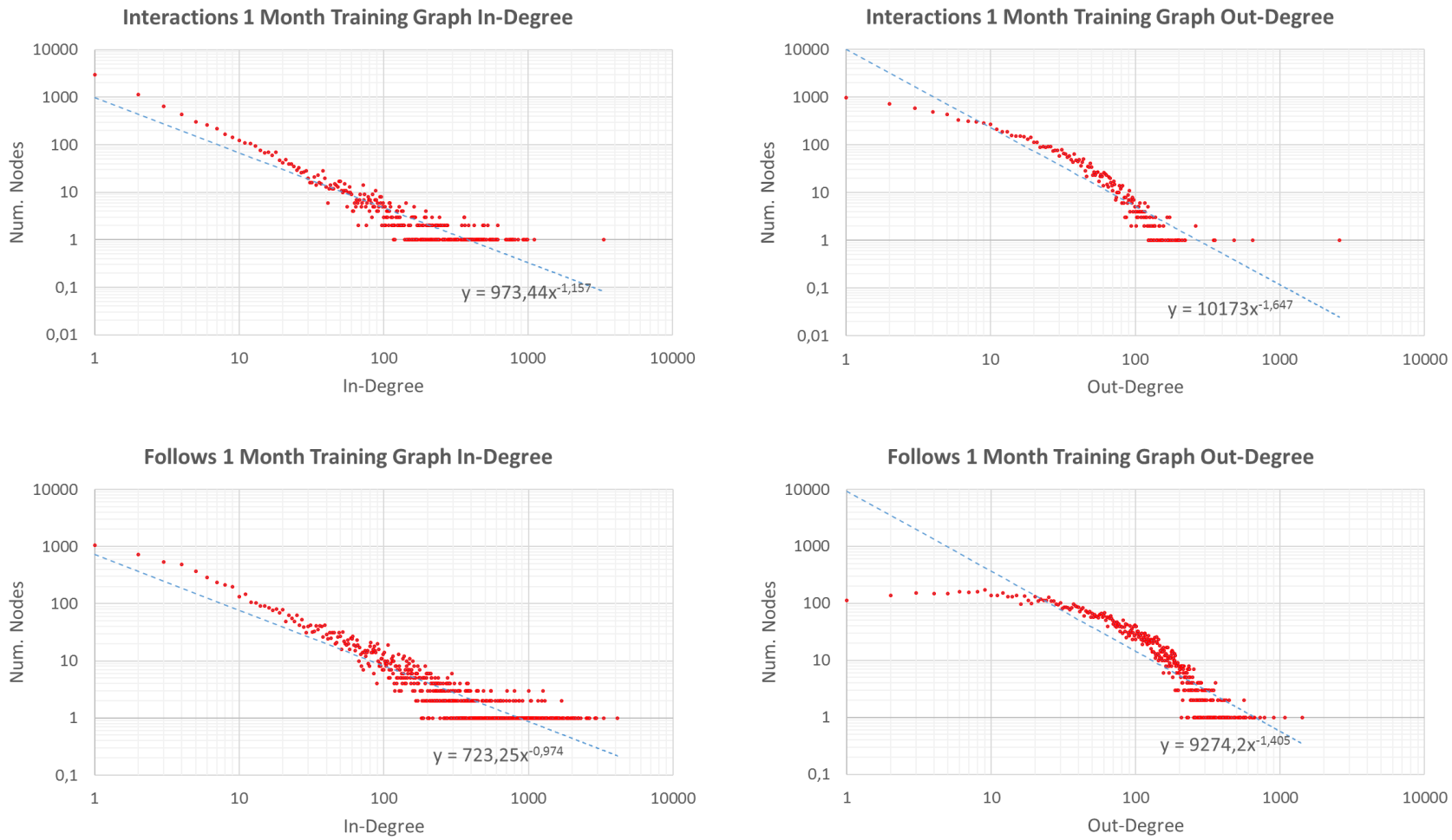


Figure 26. Training graphs degree distributions (1 Month)

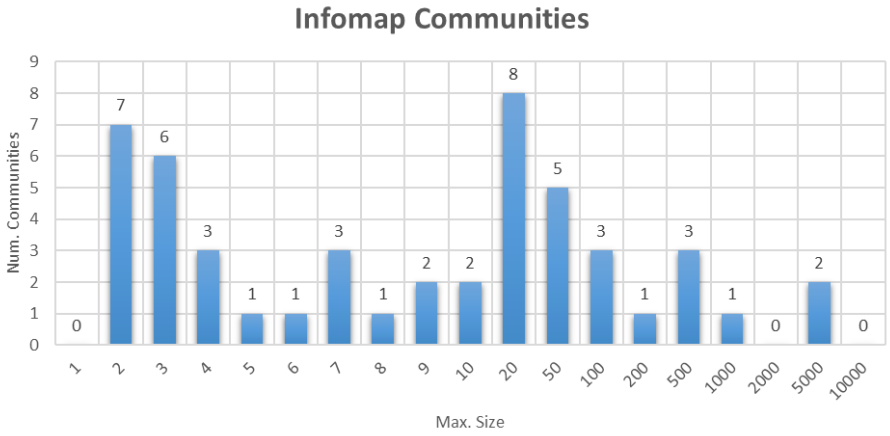


Figure 27. Infomap community distribution (1 Month follows)

200 Tweets

The second dataset expands the graph by collecting interactions from the 200 last datasets published by each one of the users (until 2nd August 2015), starting with @jesulink. It contains 10,000 different users, and 1,964,585 tweets. 341,178 of those tweets contain at least one hashtag from the 100,063 retrieved. The interactions graph (see Figure 28) has 168,128 links, extracted from a set of 482,613 interactions.668,070 interactions. mentions). The follow graph (Figure 29Figure 29) was downloaded the 20th October 2015 (training set) and 19th February 2016 (test set) and it contains a total of 475,531 links between nodes.

In this case, the selection of the date for the split was made so that the 80% of the tweets were in the training set, and the remaining 20% were in the test set. The selected date was the 29th July 2015, at 1:08:07. In Table 6, the number of links and nodes for each one of the sets is shown. From the 5,652 nodes in the training set, 4,359 have created the links.

	Nodes	**Edges**
Train	9,985	137,850
Test	5,652	21,598

Table 6. 200 Tweets interactions partition

The same data for the follows graph is shown in Table 7. 6,747 nodes have generated new links between the two download dates.

	Nodes	**Edges**
Train	9,964	427,568
Test	7,975	46,760

Table 7. 200 Tweets follows partition

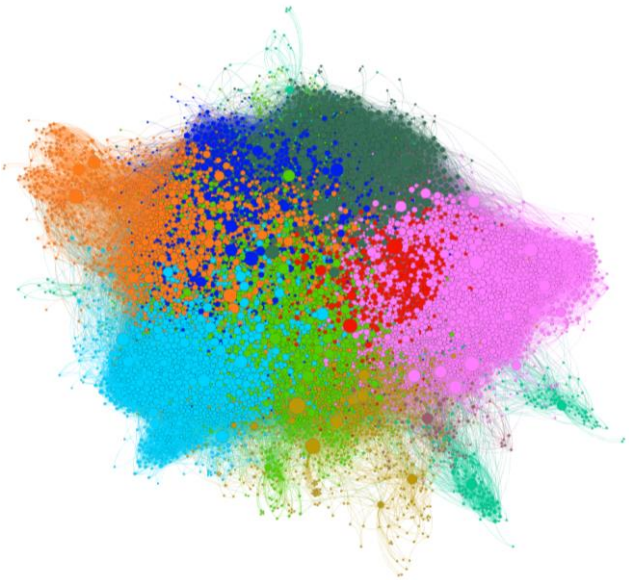


Figure 28. 200 Tweets interactions graph (Colors represent Louvain communities detected on the training graph).

Figure 29. 200 Tweets follows graph (Colors represent Louvain communities detected on the training graph)

Table 8 shows a summary of the graph metrics for both interaction and follows graph and Figure 32 shows their degree distribution. It can be noticed that the density and average degree of the follows graph is much higher than the interactions one. The diameter and the average shortest path length are larger in the interactions graph, but, surprisingly, the average eccentricity of the follows graph is higher. Both graphs have a giant component with more than the 85% of the nodes.

Metric	Interactions	Follows
Density	0.0014	0.0046
Average Degree	13.805708	42.911280
Clustering Coefficient	0.09	0.18
Average Local Clustering Coefficient	0.14	0.19
Edge Embeddedness	0.0283	0.0565
Average Shortest Path Length	4.69	3.36
Diameter	19	10
Average Eccentricity	6.36	6.50
Strongly Connected Components	1,319	1,198
Biggest SCC	86.7%	87.5%

Table 8. Training set metrics (200 Tweets)

The community metrics for the interactions graph are displayed in Table 9. Again, the number of communities detected by Infomap is higher than Louvain and Leading Vector, and the modularities of the three methods are similar. The sizes of the communities can be studied in Figure 30, Figure 31 and Figure 33.

Algorithm	Num.	Modularity	Weak Ties	In-Gini	Pair-Gini
Louvain	10	0.6048	49,668	0.5935	0.3869
Leading Vector	4	0.5029	47,226	0.8545	0.7335
Infomap	170	0.5468	64,135	0.1481	0.0272
SCC	1319	0.0198	14,963	0.2606	0.0003

Table 9. Community metrics (200 Tweets interaction graph)

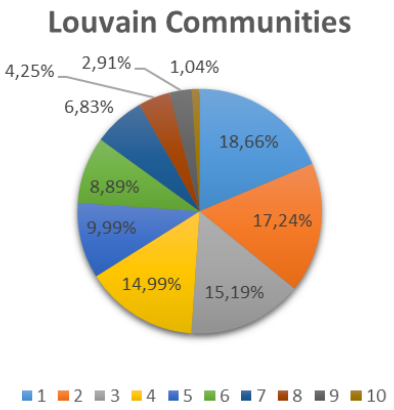


Figure 30. Louvain community sizes (200 Tweets interactions)

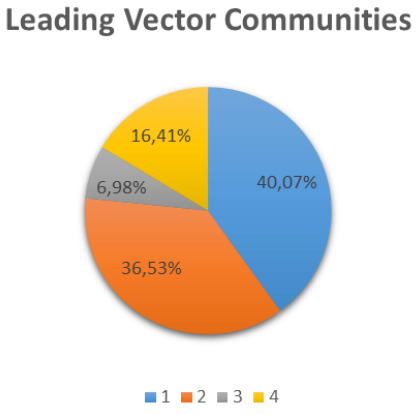


Figure 31. Leading Vector community sizes (200 Tweets interactions)

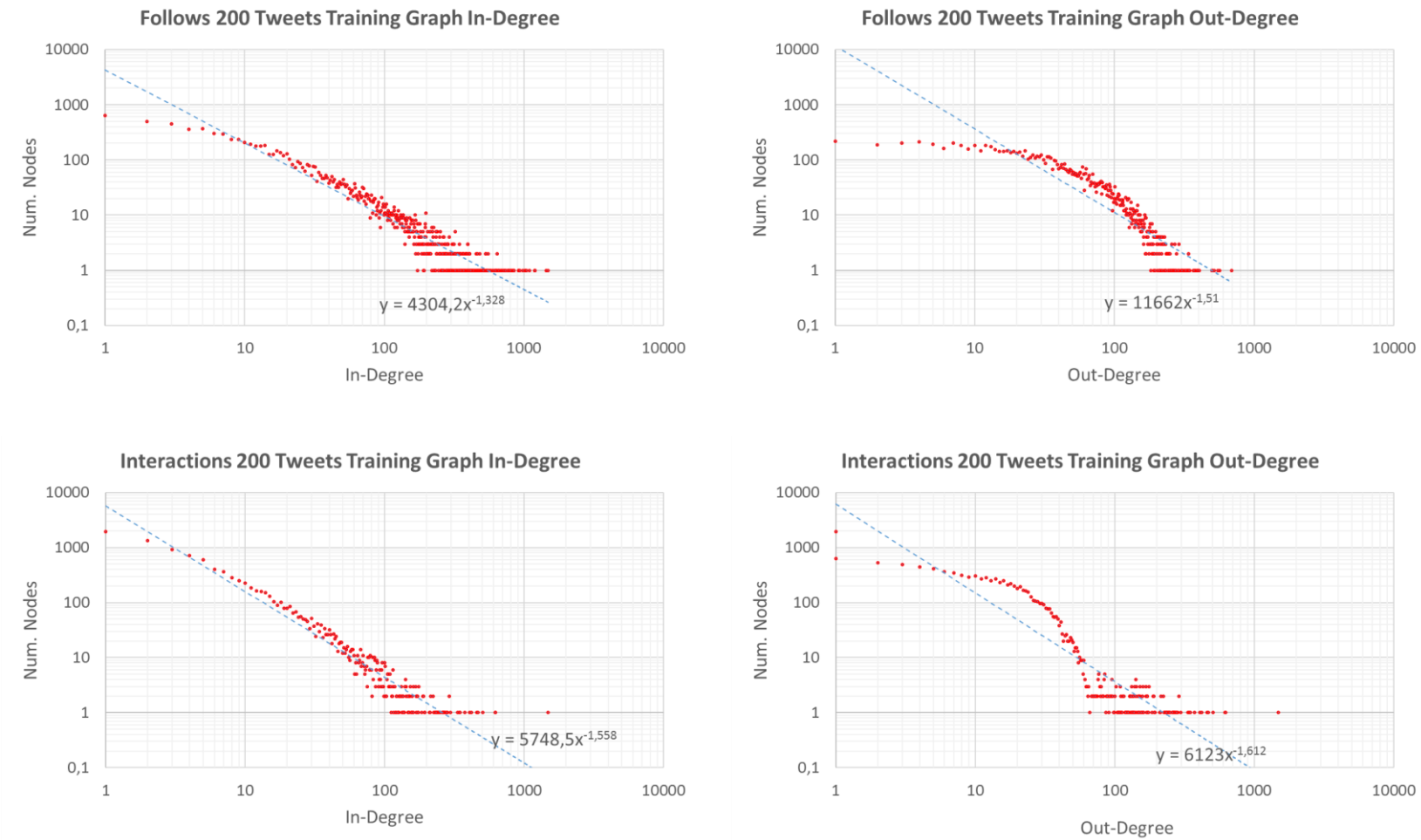


Figure 32. Training graphs degree distributions (200 Tweets)

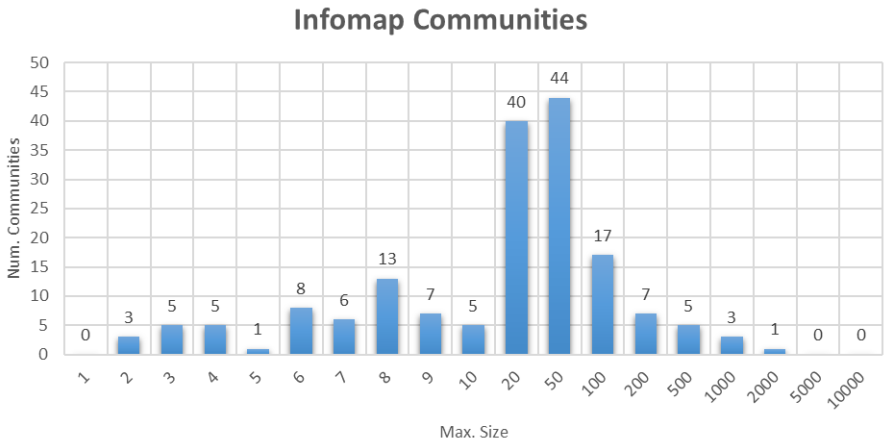


Figure 33. Infomap community distribution (200 Tweets interactions)

Algorithm	Num.	Modularity	Weak Ties	In-Gini	Pair-Gini
Louvain	9	0.5845	151,502	0.6703	0.4352
Leading Vector	3	0.5942	112,760	0.9743	0.8506
Infomap	74	0.6375	145,477	0.0947	0.0186
SCC	1,198	0.0127	30,769	0.0611	0.0003

Table 10. Community-based metrics (200 Tweets follow graph)

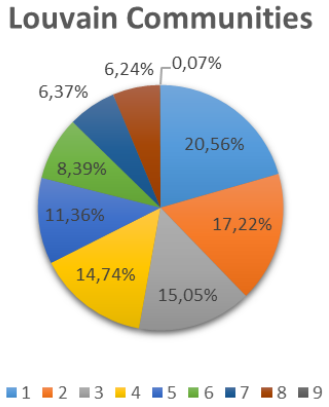


Figure 34. Louvain community sizes (200 Tweets follows graph)

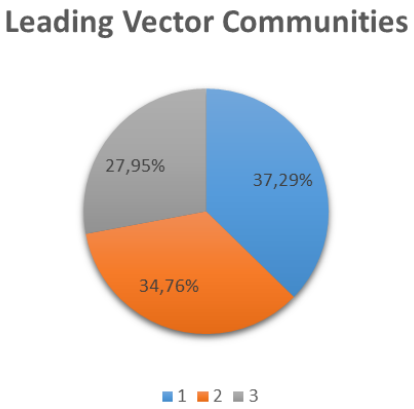


Figure 35. Leading Vector community sizes (200 Tweets follows graph)

5.2 Research Questions

The research done in the present work, and the experiments documented in the present chapter aim answering several research questions which we list next. Those questions are divided in three categories: questions related to the accuracy of the algorithms, questions related to other evaluation perspectives (novelty, diversity and structural graph diversity), and, finally, questions related to the differences between follows graphs and interaction graphs.

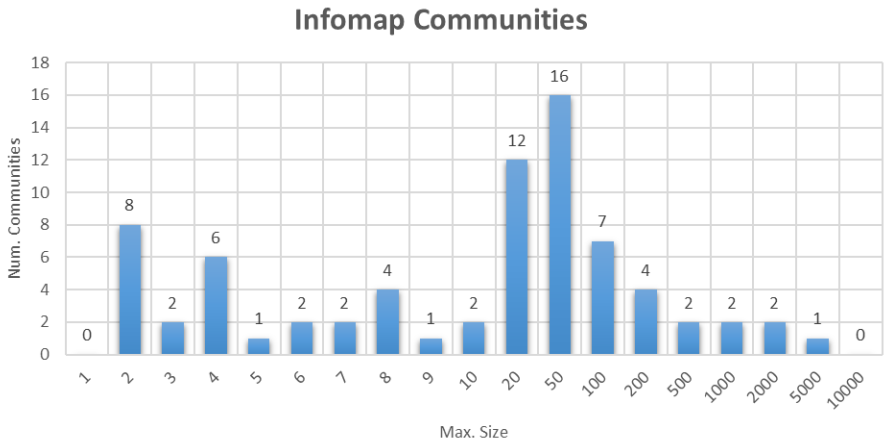


Figure 36. Infomap community size distribution (200 Tweets follows graph)

5.2.1 Accuracy Perspective

As we have stated before, the accuracy perspective focuses on maximizing the hits of the system. The questions we want to answer for this perspective are the following:

RQ1. What contact recommendation, link prediction, etc., algorithms from the literature are best in the contact recommendation task, evaluated as a ranking task on an online network such as Twitter?

RQ2. Is the adaptation of IR methods (BM25, LM, etc.) competitive in contact recommendation? And the adaptation of recommender system algorithms (HKV, kNN, etc.)?

RQ3. Are content-based methods competitive?

RQ4. How do other algorithms compare to the methods reported to be used in Twitter such as SALSA (and its WTF variant in particular)?

RQ5. In an asymmetric network, what edge direction works better for the different methods?

5.2.2 Other Evaluation Dimensions

The questions stated here refer to the rest of perspectives of evaluation: novelty, diversity, and the structural diversity metrics.

RQ6. What can novelty and diversity mean in the context of social networks and contact recommendation?

RQ7. Does the adaptation of IR (subtopic recall, ERR-IA) and recommender system (ILD, popularity complement, profile distance, Gini, etc.) novelty and diversity metrics make sense in this context? What is observed for the different algorithms?

RQ8. Do SNA metrics and concepts (weak links, bridges, clustering coefficient, redundancy, modularity, edge betweenness, target user betweenness, recommended user distance, etc.) provide any meaningful structural dimension as contact recommendation novelty, diversity or otherwise recommendation benefit metrics? (e.g. is it useful to recommend weak links?)

RQ9. What algorithms are best and worst for each metric?

RQ10. Which metrics correlate (are redundant) to each other?

RQ11. Which metrics are most and least interesting or useful?

RQ12. Are community-based metrics stable over different community finding algorithms?

5.2.3 Social Network Types

In the present work, we use two different types of graph: interactions graph and follows graph. The differences between both types of graph are the problems addressed with the following questions.

RQ13. Are the answers to all the above any different for the follow graphs and the interaction graphs?

5.3 Software configuration and test environment

To develop the experiments, we have implemented the different recommendation algorithms and metrics in the 8th version of the Java programming language. The only exception have been the different recommendation algorithms which contained matrices in their formulation (Local Path Index, Katz, Leicht-Holme-Newman Index 2, Matrix Forest, Hitting Time and Commute Time). Although they were originally implemented using Java libraries like Colt[7] or Apache Commons Math[8], we finally used Matlab for using those algorithms, since matricial operations are more efficient.

In the software we implemented for running the different experiments, we used some external libraries to prevent the full implementation of the full battery of algorithms and metrics. Here, we describe the most important ones:

- **RankSys[9]:** A public framework for the implementation and evaluation of recommendation algorithms. This library contains several recommendation algorithms as well as multiple precision, novelty and diversity metrics. We have implemented the full set of contact recommendation algorithms as an extension of this framework, using the different interfaces it provides for the development of new recommendation algorithms.
  From the already implemented methods in the framework, we have used its versions of User-Based kNN, Item-Based kNN, Random and ImplicitMF. Both precision metrics (Precision, Recall and nDCG) and the novelty and diversity ones (PC, ILD, ERR-IA…) were also computed using this framework.
- **Apache Lucene[10]:** A text search engine library written in Java. This library has been used for creating and accessing the different indexes for the Hannon algorithm (Hannon et al. 2011).
- **Gephi Graph API[11]:** A graph library which contains the implementation of Louvain algorithm (Blondel et al. 2008).
- **Criteria Comparison[12]:** Java project developed for the analysis of community mining algorithms (Rabbany et al. 2012). It contains a wrapper for the Infomap original implementation, written in C++[13], which we use in the present project.

[7] Colt: https://dst.lbl.gov/ACSSoftware/colt/ (Accessed 07/02/2017)
[8] Apache Commons Math: http://commons.apache.org/proper/commons-math/ (Accessed 07/02/2017)
[9] RankSys: http://ranksys.org (Accessed 07/02/2017)
[10] Lucene: http://lucene.apache.org/core/ (Accessed 07/02/2017)
[11] Gephi Graph API: https://github.com/gephi/gephi/wiki/Graph-API (Accessed 07/02/2017)
[12] Criteria Comparison: http://webdocs.cs.ualberta.ca/~rabbanyk/criteriaComparison/

- **Jmod[14]:** A Java toolkit for community detection in networks. It contains an implementation for the Leading Vector (Newman 2006) algorithm.

The different experiments were run in three different systems: a local computer running Microsoft Windows 7 and two different Unix servers owned by the research group. The algorithms developed in Matlab were executed over the local computer, which has 32 GB of RAM and 2 Intel Xeon E5-2620 processors at 2.40 GHz. The most part of the experiments was run in the first of the servers, which has 512 GB RAM and 64 AMD Opteron 6276 processors running at 2.3 GHz. Finally, some of the graph metrics were computed on the second server. This server has 256 GB RAM and 24 AMD Opteron 6180 processors at 2.5 GHz.

5.4 Results

In this section, we show, compare and analyze the results of our offline experiments with the full set of algorithms. In this section we only show the results for the optimal configurations of each algorithms. The different parameters have been selected using a grid search. To ease the reading of the present document, in this section we will only show the most relevant and interesting results. Full results are shown in Annex II: Complete results.

5.4.1 Accuracy

First of all, we study the accuracy of the different recommendation algorithms previously explained, in terms of the metrics of precision, recall and nDCG at cutoff 10 (we only consider the top 10 recommended users). In tables 11, 12, 13 and 14, we show the P@10 values for the top 10 and the top bottom algorithms for each one of the graphs. The color gradient for each P@10 column shows how good the precision value in comparison to the rest of the algorithms: best results are shown in green, and the worst ones are colored in red. Along the name of each algorithm, we show the configuration we have used, previously selected through a grid search.

Although there are many differences between the top 10 algorithms for the four studied graphs, we also observe some similarities between them. In particular, there is a family of algorithms which is present in all four rankings: the adapted IR algorithms based in the probability ranking principle: BIR, BM25 and Extreme BM25. The selection of the ten bottom algorithms seems to be rather consistent in the different graphs: the random baseline algorithm and the different variants of probabilistic matrix factorization (PMF Basic and PMF Sigmoid) are always among the four worst recommendation algorithms. The different Leicht-Holme-Newman indexes are also among the worst algorithms, as well as the distance-based recommenders.

After observing those differences and similarities, we raise the following question: are the rankings for the different datasets really different? To solve that, we show a comparison between the interactions and the follows graphs for each dataset in figures 37 and 38, we compare the two interactions graphs in Figure 39, and, finally, we compare the two follows graphs in Figure 40. In those graphs, we observe that both graphs in the 200 Tweets dataset are very similar in terms of ranking, and the same (in lesser extent) happens between the two interactions graphs. The only graph which provides very different results is the follows graph in the 1 Month dataset.

[13] Infomap: http://www.mapequation.org/code.html
[14] Jmod: http://tschaffter.ch/projects/jmod/

Interactions 1 Month	
Algorithm	**P@10**
ImplicitMF (k=280,α=150;λ=40)	**0,0612**
UserBased kNN (k=120)	0,0598
Personalized SALSA (Authorities;r=0.99)	0,0577
Average Cosine Similarity	0,0554
ItemBased kNN (k=300)	0,0544
Centroid Cosine Similarity	0,0541
Local Path Index (Und.;β=0.1;l=3)	0,0530
Adamic/Adar $\left(\Gamma_{und}(u);\Gamma_{in}(v);\Gamma_{und}(w)\right)$	0,0506
BIR ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0,0505
Money (k=1000;α=0.3)	0,0485
Hub Depressed Index ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0,0169
TF-IDF ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0,0139
Salton ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0,0127
Katz (Dir.;β=0.1)	0,0102
Distance (Dir.)	0,0063
LHN Index 1 ($\Gamma_{in}(u);\Gamma_{out}(v)$)	0,0032
PMF Sigmoid (k=50.0;λ=0.01;ϕ=0.01)	0,0024
LHN Index 2 (β=0.4)	0,0012
Random	0,0006
PMF Basic (k=40.0;λ=0.01;ϕ=0.01)	0,0005

Table 11. P@10 for the best and worst algoritms (Interactions 1 Month)

Follows 1 Month	
Algorithm	**P@10**
ExtremeBM25 ($\Gamma_{und}(u);\Gamma_{und}(v)$;b=0.99)	**0.0067**
BM25 ($\Gamma_{und}(u);\Gamma_{und}(v)$;b=1.0;k=1000.0)	**0.0067**
PageRank (r=0.6)	0.0032
Personalized PageRank (r=0.5)	0.0030
Popularity	0.0028
PrefAttach (IN)	0.0028
Local Path Index (Und.;β=0.3;l=5)	0.0027
Katz (Dir.;β=0.9)	0.0020
Pure Personalized PageRank (r=0.5)	0.0019
Resource Allocation $\left(\Gamma_{und}(u);\Gamma_{in}(v);\Gamma_{und}(w)\right)$	0.0017
Hub Depressed Index ($\Gamma_{in}(u);\Gamma_{und}(v)$)	0.0003
Average Cosine Similarity	0.0003
Matrix Forest (α=0.01)	0.0002
Distance (Und.)	0.0001
LHN Index 1 ($\Gamma_{in}(u);\Gamma_{und}(v)$)	0.0001
LHN Index 2 (β=0.1)	0.0001
PMF Sigmoid (k=50;λ=0.01;ϕ=0.01)	0.0000
Maximum Cosine Similarity	0.0000
PMF Basic (k=50;λ=0.01;ϕ=0.01)	0.0000
Random	0.0000

Table 12. P@10 for the best and worst algorithms (Follows 1 Month)

Interactions	
Algorithm	**P@10**
BM25 ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$;b=0.3;k=1.0)	**0,0238**
ExtremeBM25 ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$;b=0.1)	0,0237
BIR ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$)	0,0233
Adamic/Adar$\left(\Gamma_{und}(u);\Gamma_{in}(v);\Gamma_{und}(w)\right)$	0,0233
ImplicitMF (k=300;λ=150;α=40)	0,0226
FOAF (UND;IN)	0,0221
QLJM (UND;IN;λ=0.1)	0,0210
RA ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$;$\Gamma_{und}(w)$)	0,0209
Personalized SALSA (Auth.;α=0.99)	0,0209
User-Based kNN (k=100)	0,0202
Local Path Index (;β=0.1;l=3)	0,0057
Distance (Dir.)	0,0056
Commute Time	0,0054
LHN Index 1 (IN;OUT)	0,0054
Hitting Time	0,0051
LHN Index 2 (β=0.1)	0,0034
Katz (Dir.;β=0.1)	0,0032
PMF Sigmoid (k=20;λ=0.01;ϕ=0.01)	0,0006
PMF Basic (k=60;λ=0.01;ϕ=0.01)	0,0002
Random	0,0002

Table 13. P@10 for the best and worst algorithms (Interactions 200 Tweets)

Follows	
Algorithm	**P@10**
ImplicitMF (k=300;α=40;λ=150)	**0.0365**
User-based kNN (k=40)	0.0343
Average Cosine Similarity	0.0338
Adamic (UND;IN;UND)	0.0328
QLJM (UND;IN;λ=0.1)	0.0328
BM25 (UND;IN;b=0.1;k=1)	0.0326
BIR (UND;IN)	0.0323
Item-based kNN (k=300)	0.0322
Resource Allocation (UND;IN;UND)	0.0322
ExtremeBM25 (UND;IN;b=0.1)	0.0321
Distance (Dir.)	0.0032
Katz (Dir.;β=0.4)	0.0029
LHN Index 2 (β=0.2)	0.0028
Hitting Time	0.0027
Commute Time	0.0027
Personalized HITS (Auth.;α=0.99)	0.0016
Love(Auth.;k=1000;α=0.2)	0.0015
PMF Basic (k=20.0;λ=0.01;ϕ=0.01)	0.0007
Random	0.0005
PMF Sigmoid (k=20.0;λ=0.01;ϕ=0.01)	0.0004

Table 14. P@10 for the best and worst algorithms (Follows 200 Tweets)

After observing those differences and similarities, we raise the following question: are the rankings for the different datasets really different? To solve that, we show a comparison between the interactions and the follows graphs for each dataset in figures 37 and 38, we compare the two interactions graphs in Figure 39, and, finally, we compare the two follows graphs in Figure 40. In those graphs, we observe that both graphs in the 200 Tweets dataset are very similar in terms of ranking, and the same (in lesser extent) happens between the two interactions graphs. The only graph which provides very different results is the follows graph in the 1 Month dataset.

Observing those comparisons, as well as the complete results, we have detected several groups of algorithms which obtain similar precision values for all graphs. and the whole data, we have detected some groups of algorithms which obtain similar precision values in all datasets:

- **Popularity and Preferential Attachment:** Those two algorithms usually obtain the same values for the three accuracy metrics. In this case, there is no doubt about why this happens: usually, the best configuration for the Preferential Attachment algorithm consists on recommending those users with the greatest incoming neighborhood (the same as popularity-based recommendation).
- **Jaccard and Sørensen:** Those algorithms have tied in the three precision metrics in all four graphs. In this case, they are variants of the FOAF algorithm, with very similar formulations, as it is shown in equations 3.6 and 3.7. That may explain their high correlation.
- **Adamic/Adar and Binary Independent Retrieval**: In this case, the differences between both algorithms are greater than the previous groups, but we have observed that they are always near in the rankings, attaining very similar precision values. Observing their equations (3.13 for Adamic/Adar and 3.80 for BIR), we realize that both methods have similar formulations: they select those users which share many impopular neighbors with the target user, and they penalize the popularity of the common neighbors using logarithms, which explains their similarity.

Now, we will study the accuracy for the different kinds of algorithms we defined in chapter 3. For each family of algorithms, we show bar diagrams which compare the P@10 metric for each algorithm in the family (colored in blue) on each graph. Aditionally, we have added to the comparison the two baseline algorithms, random recommendation and popularity-based recommendation, which are represented user red bars, and finally, the best approach for each graph colored in green

Link prediction methods

Since this one is the most populated family, we have subdivided the analysis in three subfamilies, equally to the subdivision we made in chapter 3: neighborhood-based, path-based and random walk-based methods:

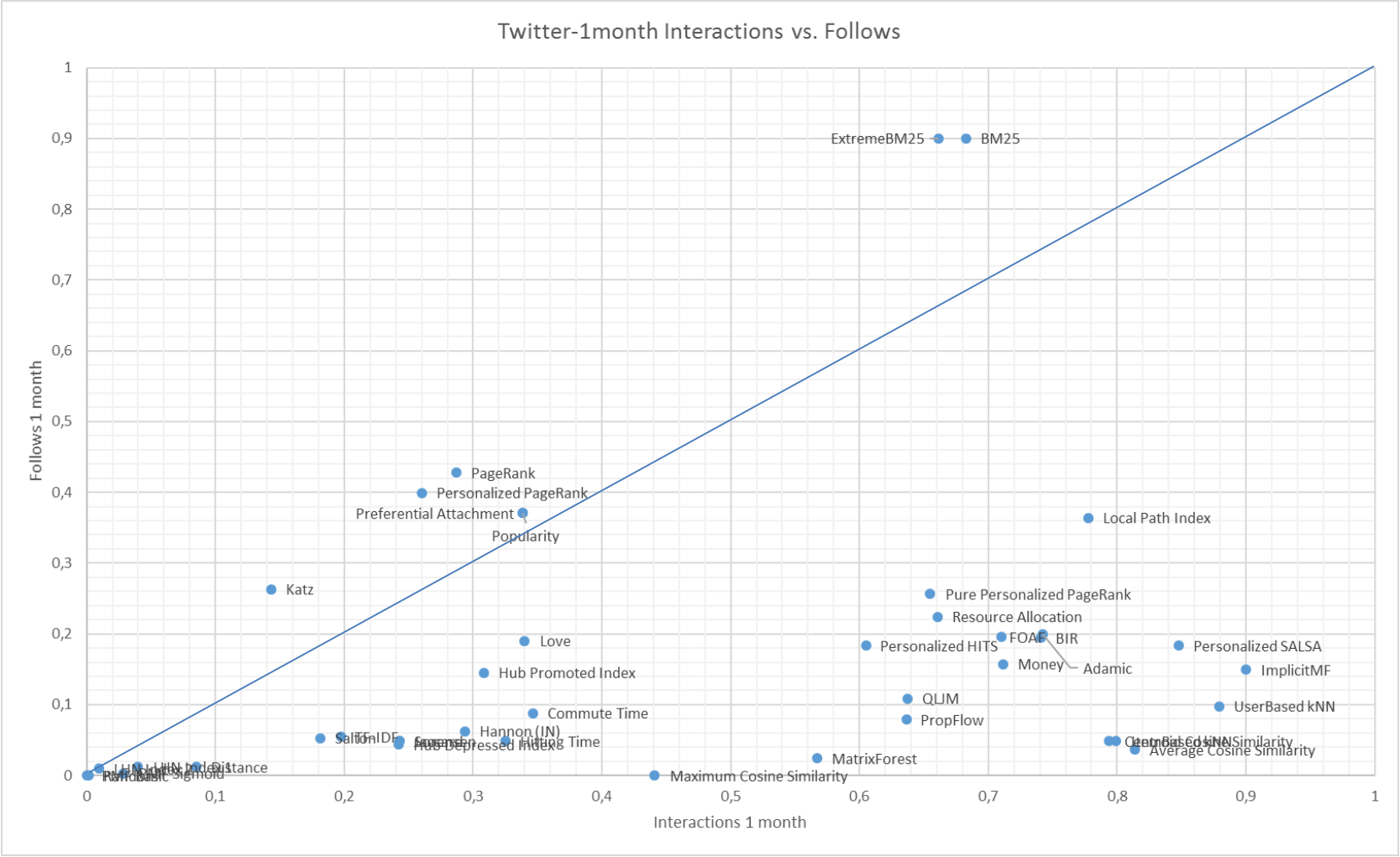


Figure 37. Comparison between Interactions 1 month and Follows 1 month algorithm rankings

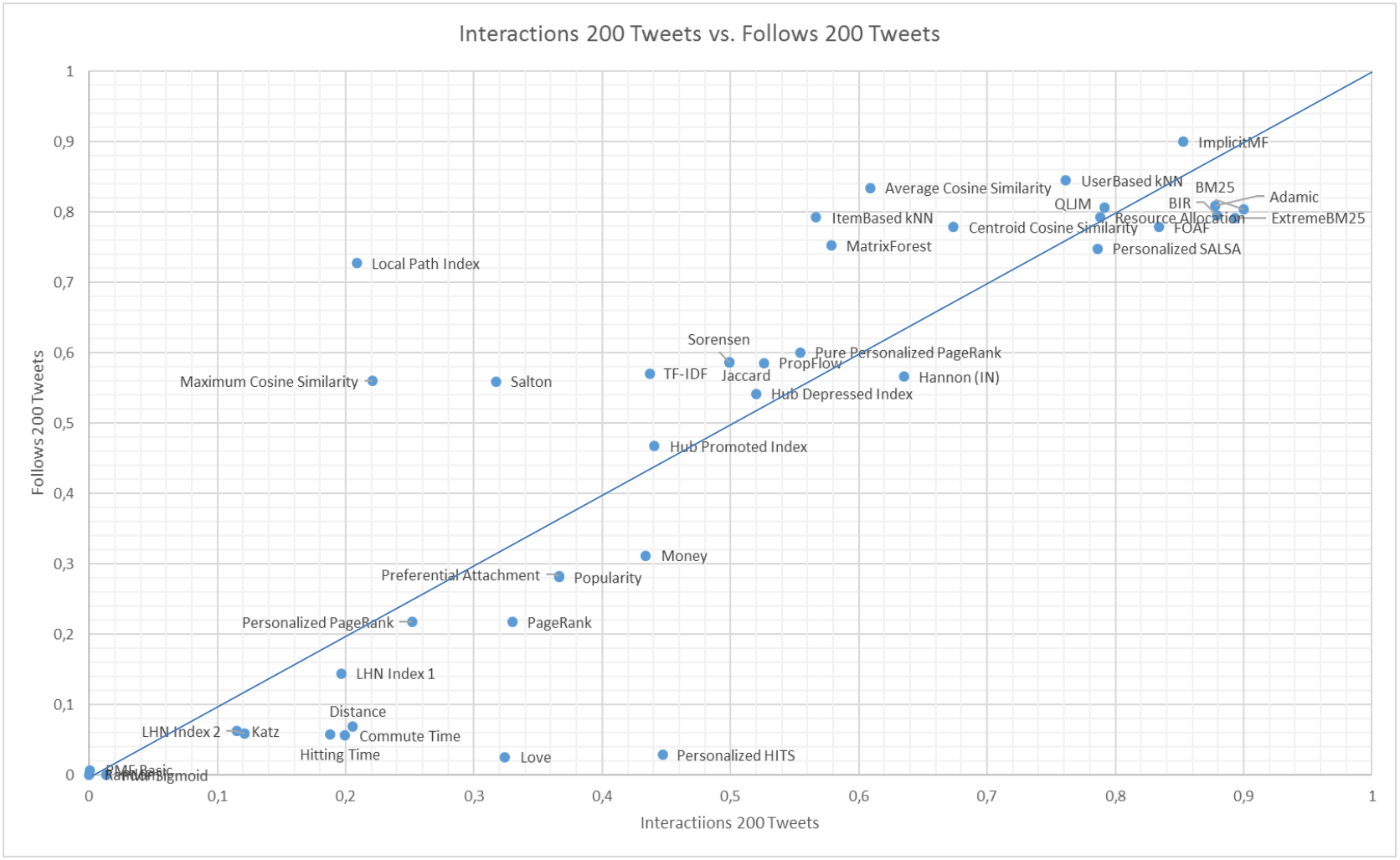


Figure 38. Comparison between Interactions 200 Tweets and Follows 200 Tweets algorithm rankings

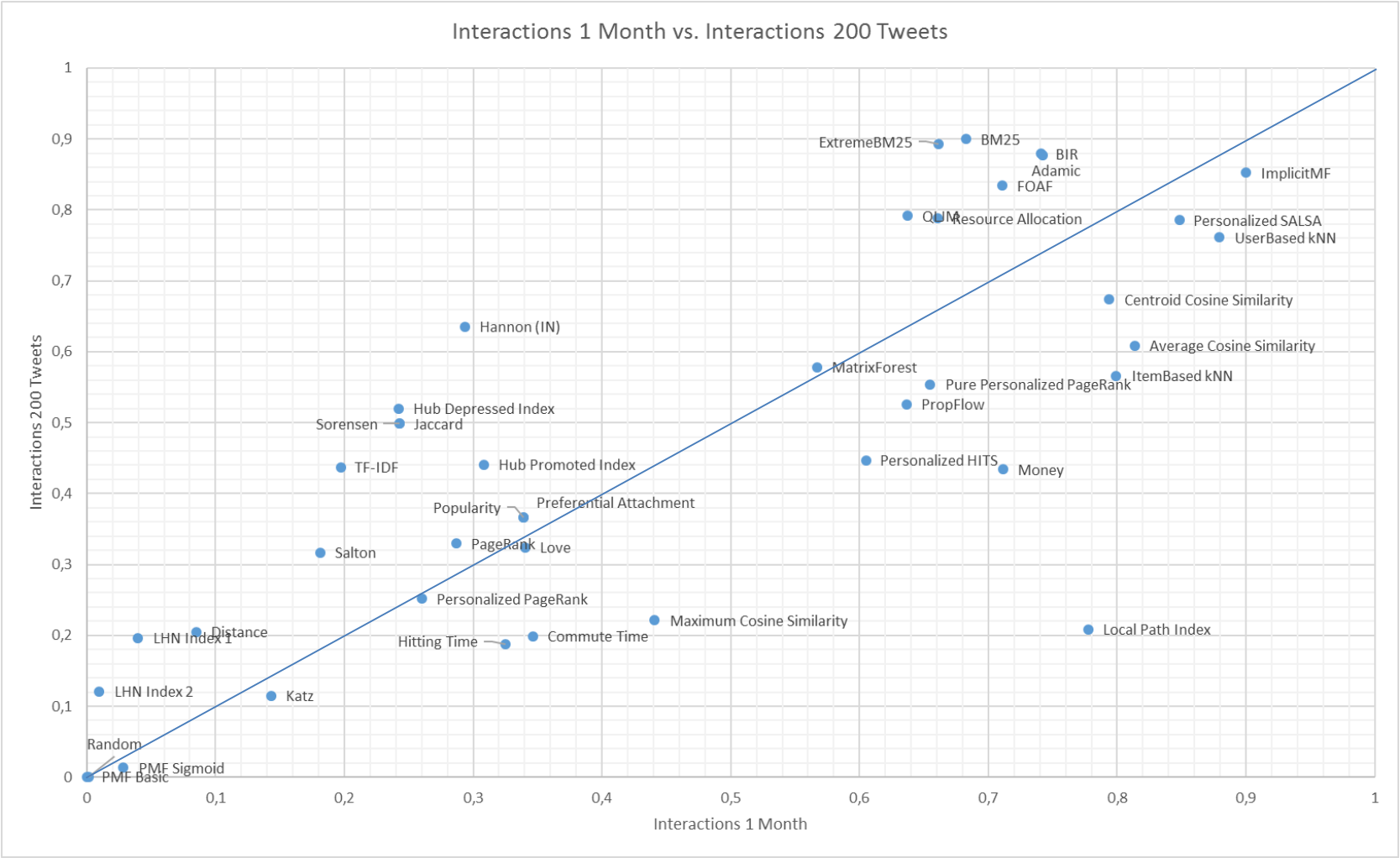


Figure 39. Comparison between Interactions 1 Month and Interaction 200 Tweets algorithm rankings

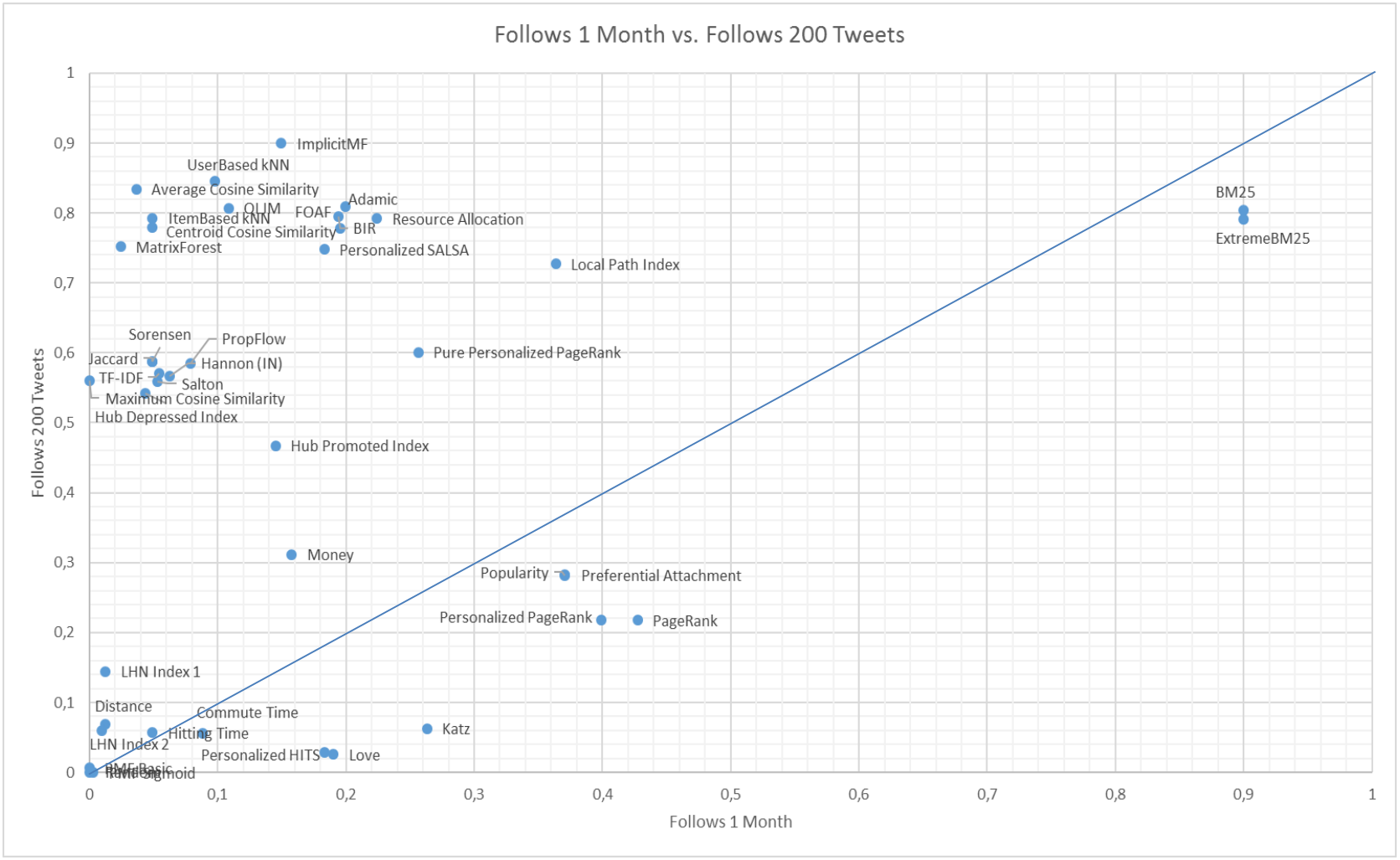


Figure 40. Comparison between Interactions 1 Month and Interaction 200 Tweets algorithm rankings

- **Neighborhood-based methods:** In this family, we identify three methods which attain competitive values for the different graphs (with the exception of the follows graph for the 1 Month dataset: Adamic/Adar, Resource Allocation, and the not normalized version of the most common neighbors algorithms. From the three algorithms, Adamic/Adar is the most effective of the three approaches, always near the best algorithms in the comparison. This can be seen in Figure 41.

  An interesting observation is that the accuracy of the normalized versions of the most common neighbors recommendation methods falls far behind the unnormalized approach. As a particular example of that, the worst approach of this family, the first Leicht-Holme-Newman index, is usually among the ten worst algorithms, while the original approach is among the fifteen best in all four graphs.

- **Path-based methods:** The accuracy of most part of the methods in this family is usually far from the optimal values achieved by other algorithms like BM25 or ImplicitMF. There are two algorithms which reach higher values for the precision than the rest: Matrix Forest and Local Path Index. However, we cannot say that both algorithms are always competitive nor establish an order between them, at least, in terms of P@10. We display the comparative between these algorithms in Figure 42.

- **Random walk-based methods:** Figure 43 shows the comparative between the different approaches in this family. We observe all the algorithms obtain low precision values, generally below the values of the popularity-based recommendation baseline, or very close to it. Only two methods clearly overtake that baseline, our novel personalized PageRank approach, Pure Personalized PageRank, and the PropFlow algorithm. We note that the classical Personalized PageRank method does not achieve good values, being even worse than the not personalized version of the algorithm.

Twitter Who-To-Follow algorithm

The comparison between these algorithms has a special interest in our investigation, since the different methods in this family are similar to the ones used by Twitter in their Who-To-Follow approach. We show them in Figure 44.

Observing that figure, we notice that the personalized version of SALSA, and two of the cosine similarity approaches, Average and Centroid Cosine Similarity, work very well in three of the four studied graphs. In fact, the Personalized SALSA algorithm, which extends the Money algorithm used in the real network, is always among the top half of the algorithms in the comparative (in the top 15 if we do not consider the follows graph for the 1 Month dataset). However, in all the different graphs, we have found several algorithms which attain better accuracy values than all the methods in this family, like the classical recommendation techniques or some of the IR adapted algorithms.

An interesting observation is that the effectiveness of the Money and Love algorithms falls behind the versions which creates the bipartite graph using all nodes in the network (Personalized SALSA and Personalized HITS, respectively), showing that, at least for this network, it is not necessary to compute the so-called circle of trust to obtain better values for the recommendation.

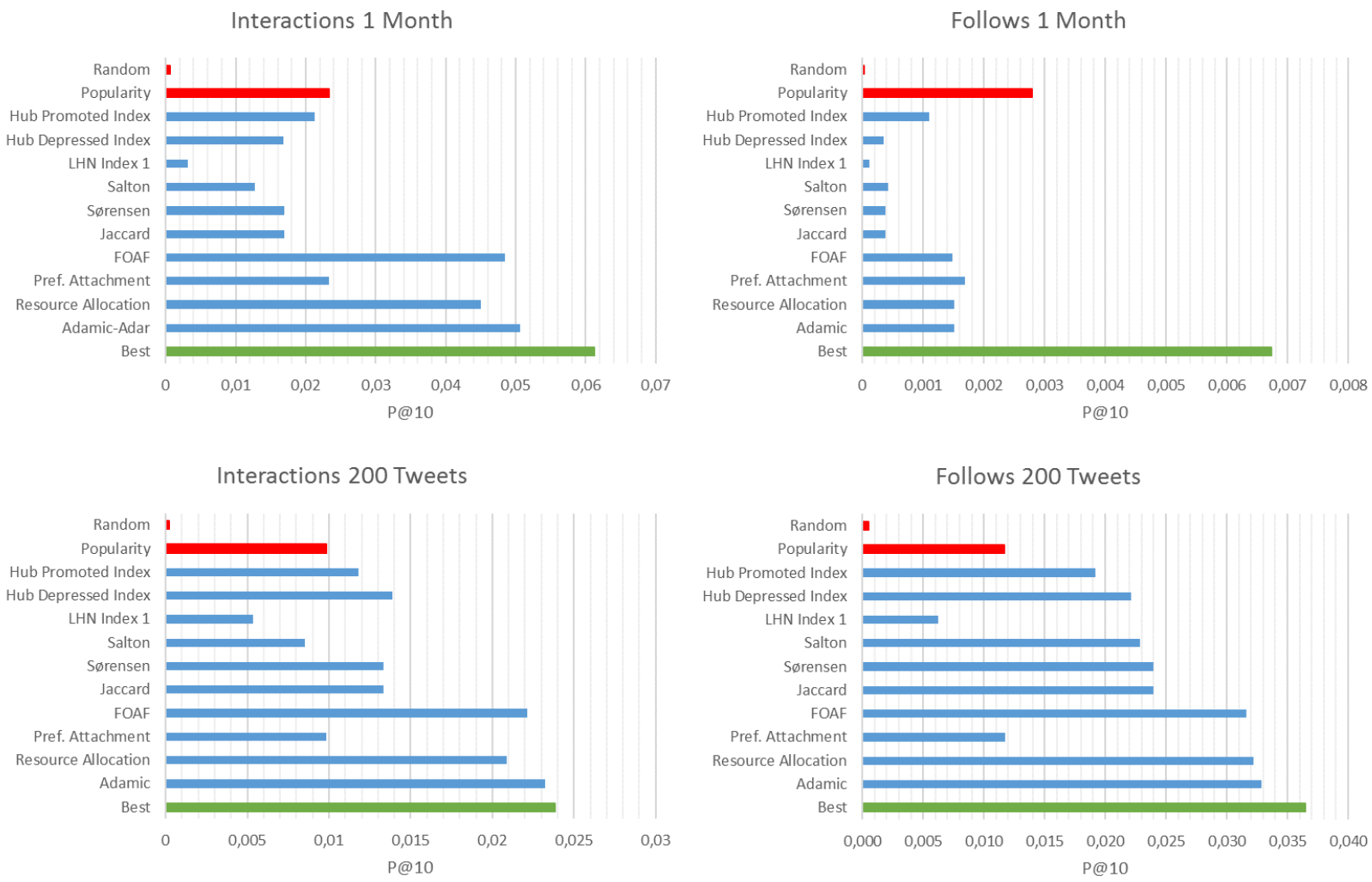


Figure 41. P@10 for the neighborhood-based link prediction methods

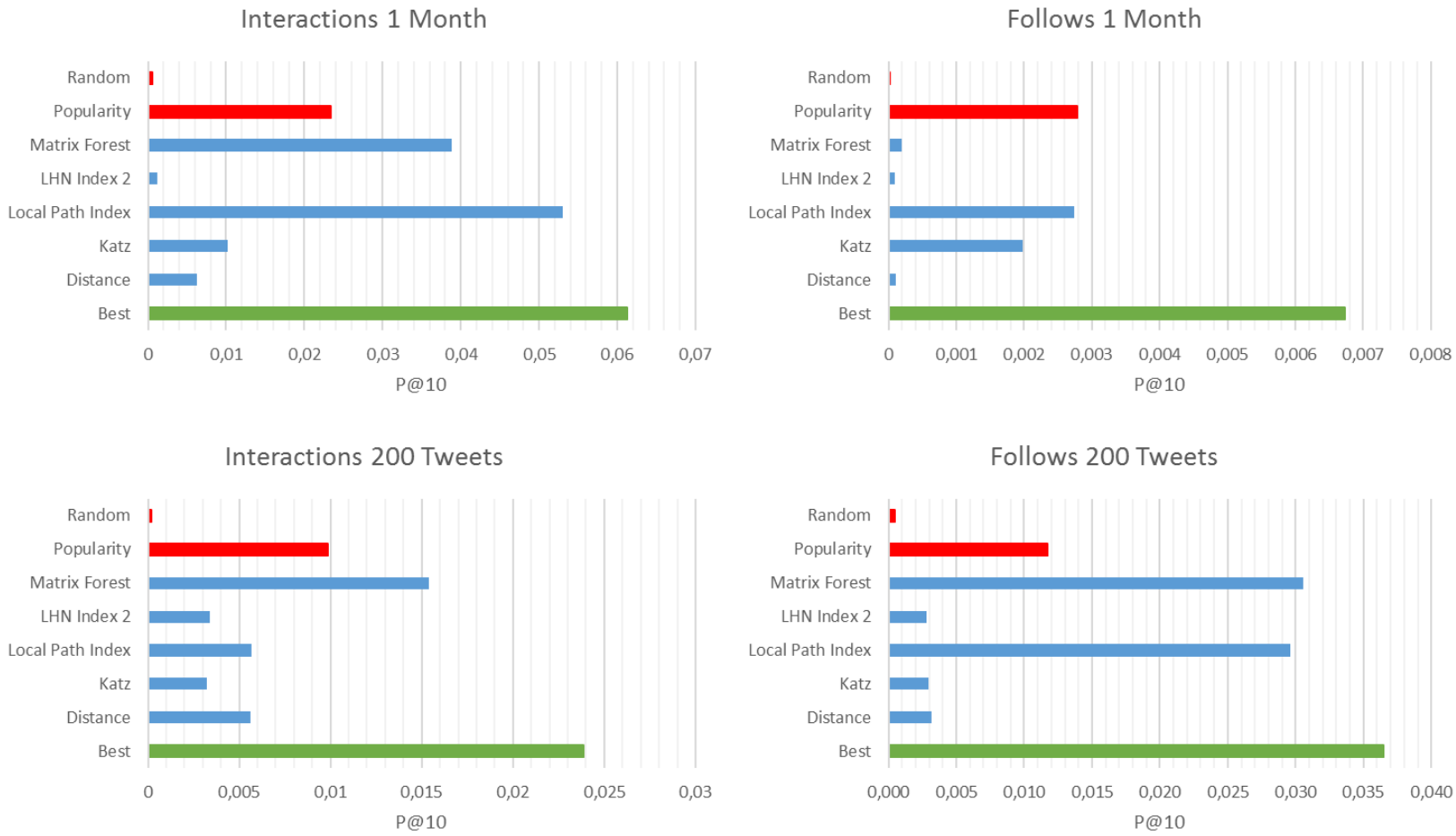


Figure 42. P@10 for the path-based link prediction methods

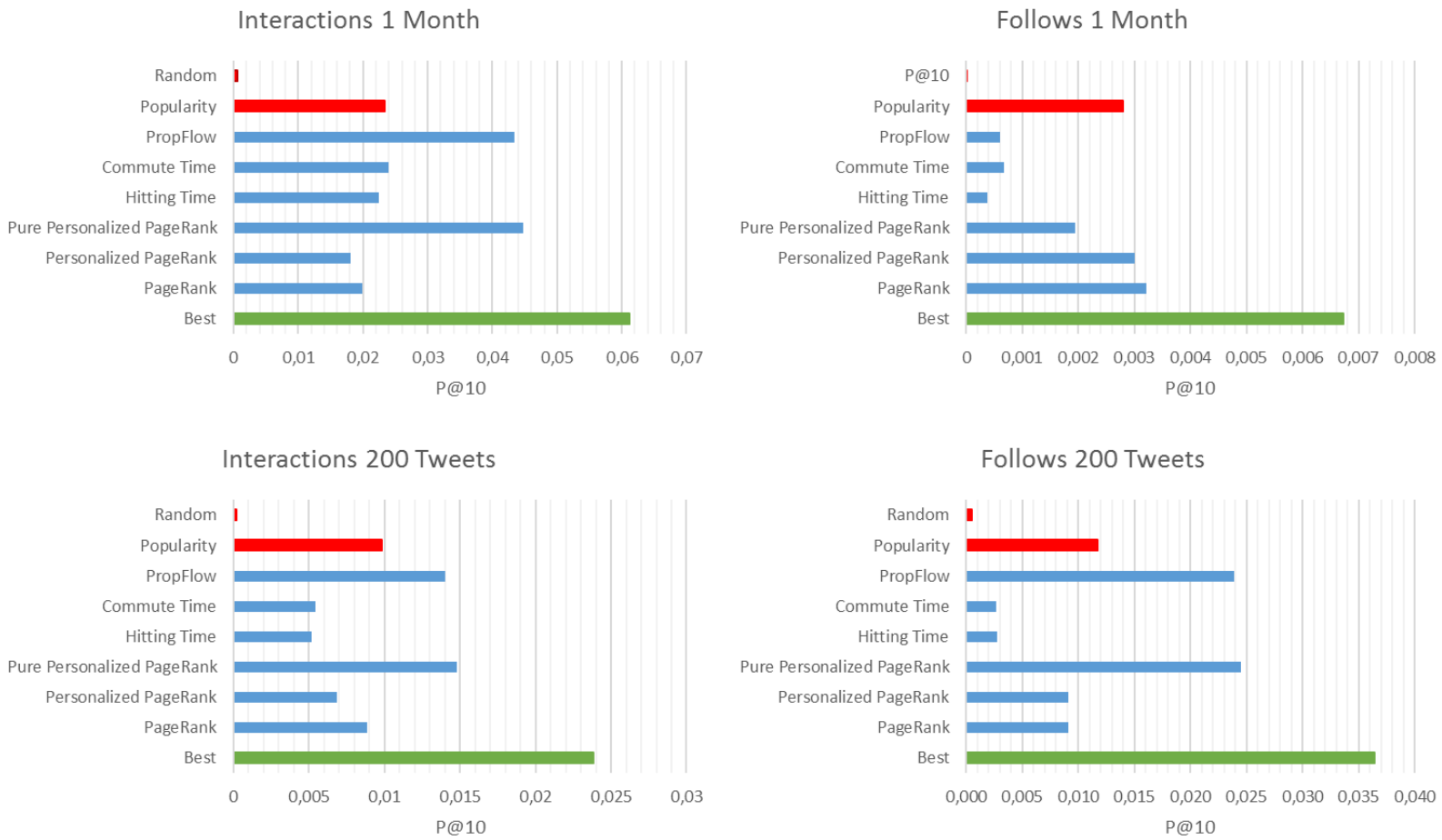


Figure 43. P@10 for the random-walk base link prediction methods

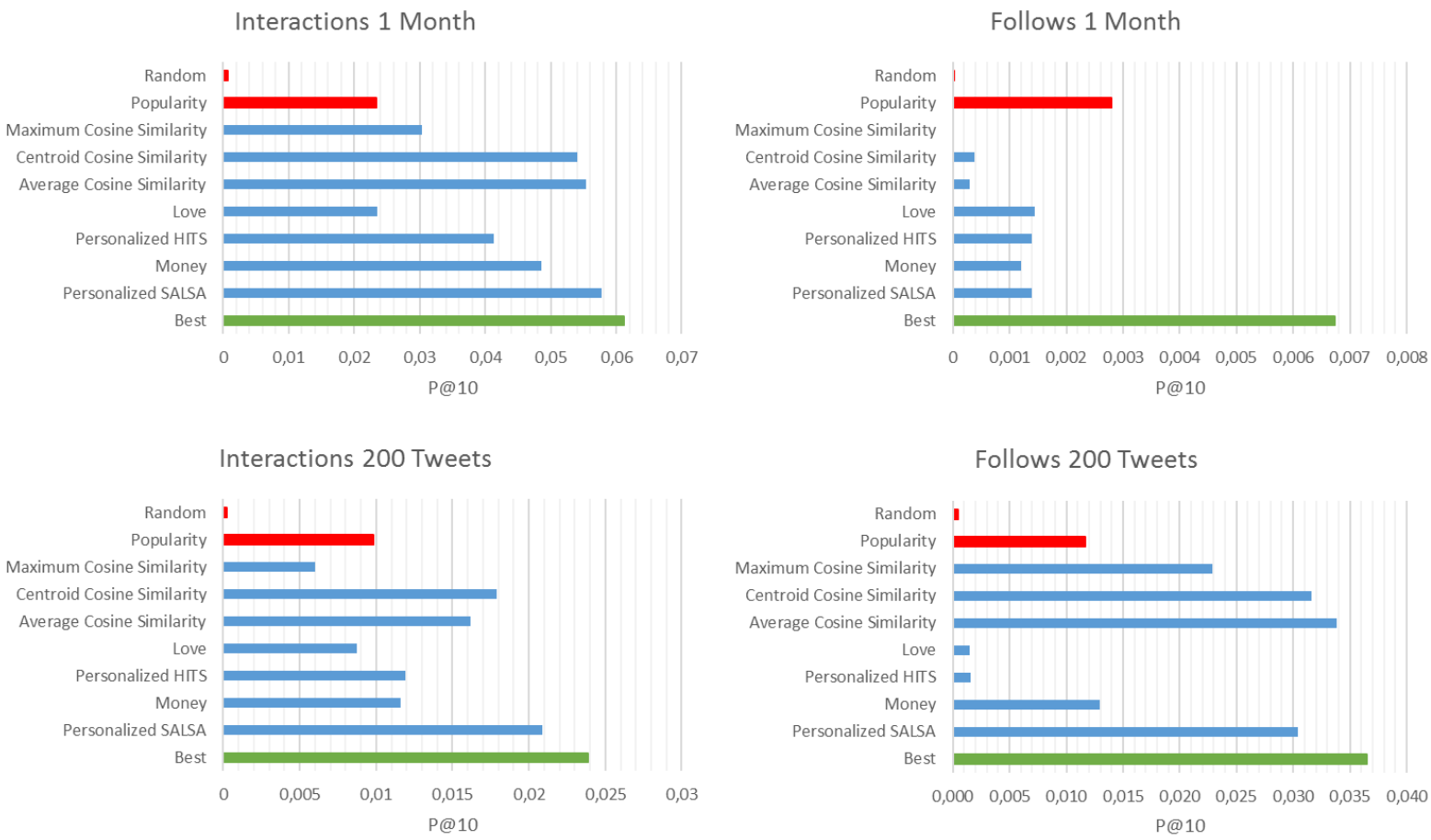


Figure 44. P@10 for the Twitter Who-To-Follow methods

Classical recommendation

The family of classical recommendation algorithms we adapt for the contact recommendation problem tains some of the best algorithms in the comparison, as well as some of the worst. In Figure 45, we show the full comparison between the different algorithms.

On one hand, it is interesting to notice that both neighborhood-based recommendation algorithms attain good results in the comparative for all graphs excepting the follows graph for the 1 month dataset, specially the user-based approach. If we observe the matrix factorization approaches, we find that the matrix factorization algorithm for implicit feedback (ImplicitMF) overtakes both neighborhood-based approaches in all

four graphs, and it even reaches the best value for any algorithm in two of them. In those graphs, user-based kNN has the second best P@10 value of all.

On the other, both PMF approaches obtain very poor results, even worse than the random recommendation baseline.

Text Information Retrieval methods

As one of our original contributions to the contact recommendation area consists on the adaptation of Text Information Retrieval algorithms like BM25 or Query Likelihood, it is important to analyze their competitiveness. As we stated before, BIR, BM25 and Extreme BM25 stand out from the rest of algorithms in our study, since at least one of them appears in the top 10 algorithms for every graph. That fact can be also observed in Figure 46, where those algorithms are always near the best recommender in the dataset (if they are not the best themselves).

The other probabilistic method, Query Likelihood, although it usually falls behind the three previously mentioned approaches, it is also highly competitive, and achieves similar results to the other ones. However, TF-IDF, the algorithm based on the vector model, is far from the rest of algorithms in this family.

Content-based recommendation (Hannon)

Finally, we will study the content-based approaches. All the algorithms we have adapted and proposed for the present work may be classified under the collaborative filtering algorithms tag, since all of them use information from other users preferences, with a single exception: the algorithm we have called Hannon. This algorithm generates characteristics vectors for each user in the network using the different tweets in the dataset.

We show the comparative for the different Hannon variants in Figure 47. In that graphic, we observe that, even when the results are far from the worst algorithms, they are also far from the best ones. For all the four graphs, we also notice that the best version of this algorithms is always the version which uses the tweets from the incoming neighbors of the users to generate the word vectors.

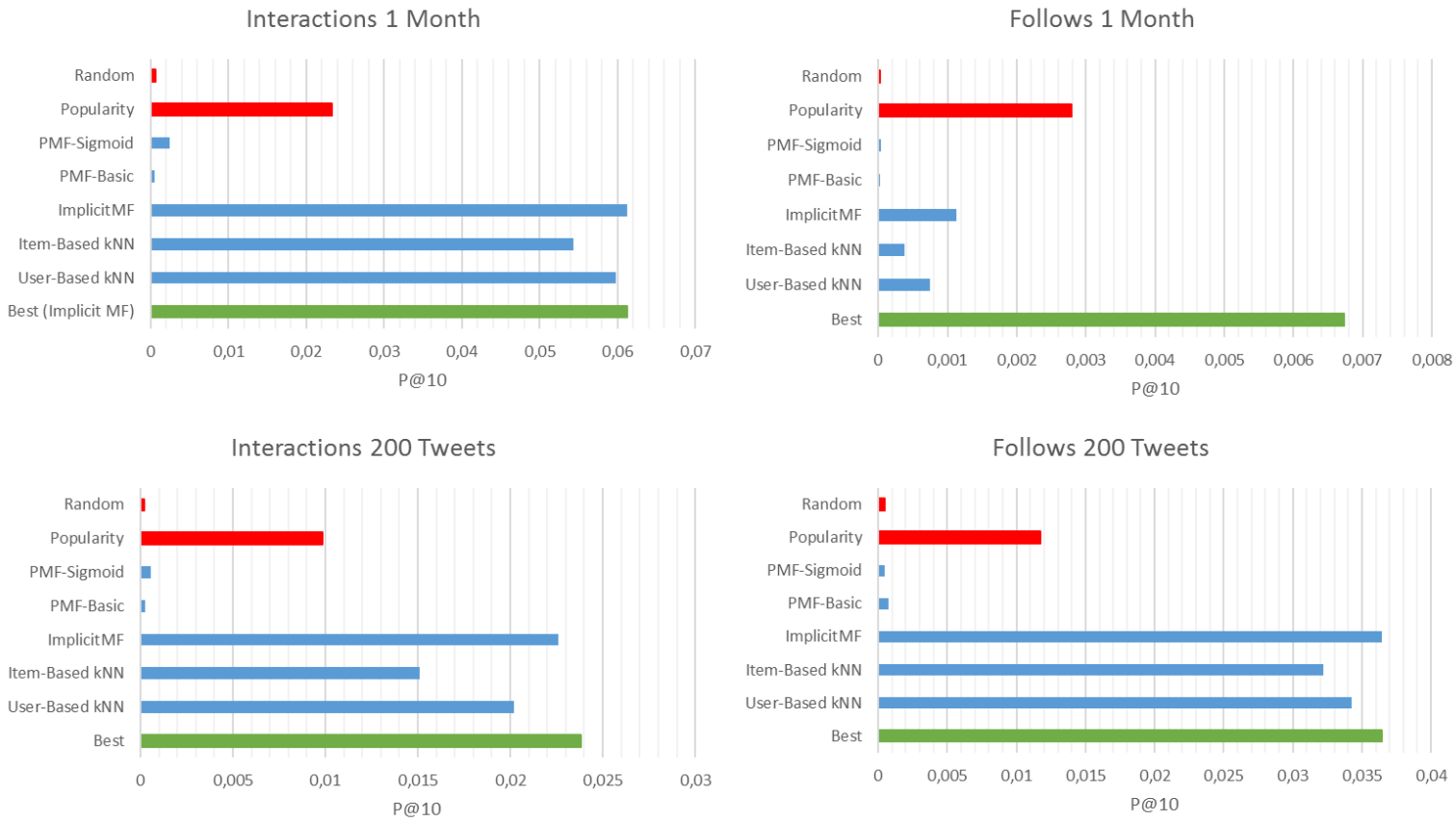


Figure 45. P@10 for the classical recommendation algorithms

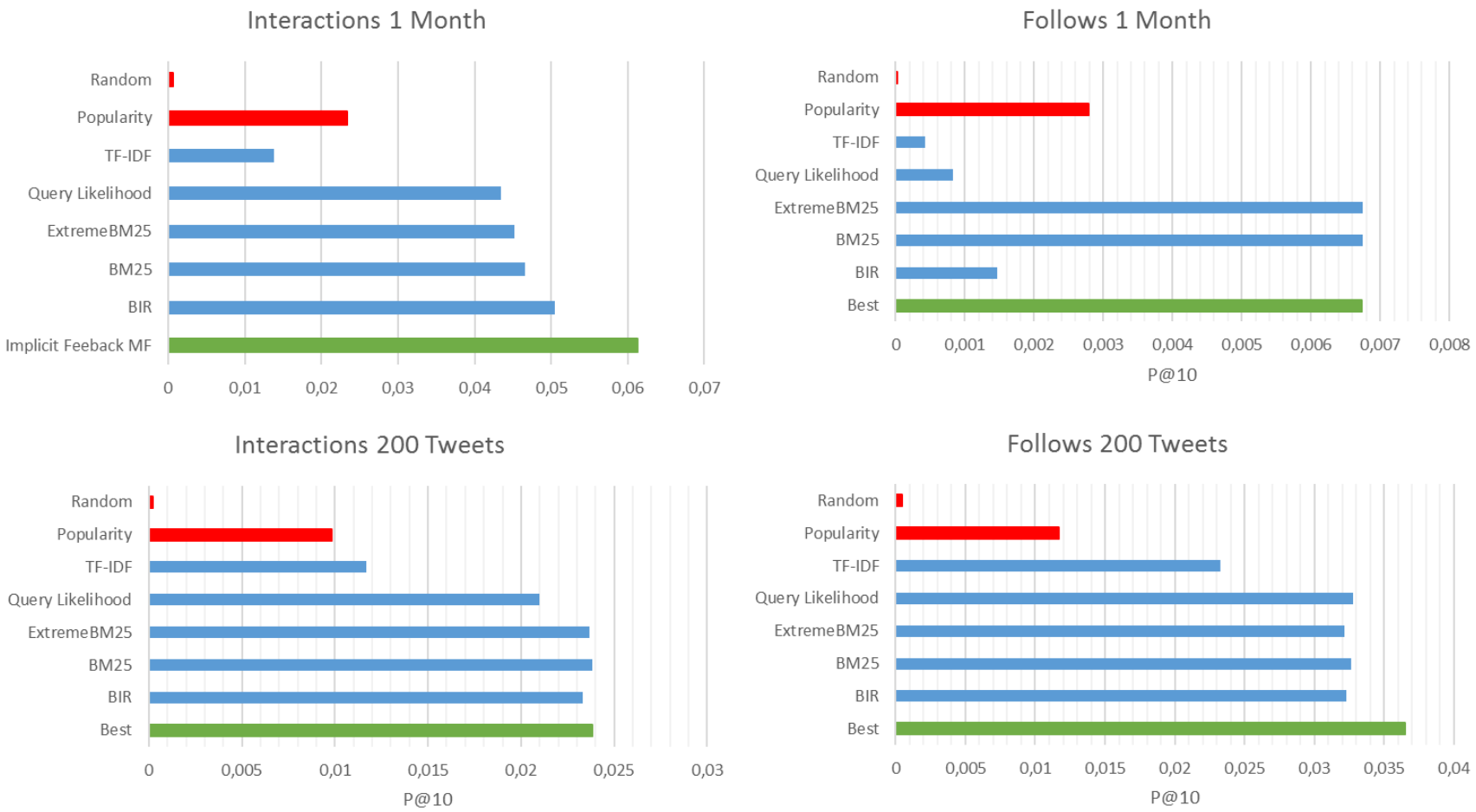


Figure 46. P@10 for the adaptations of IR algorithms

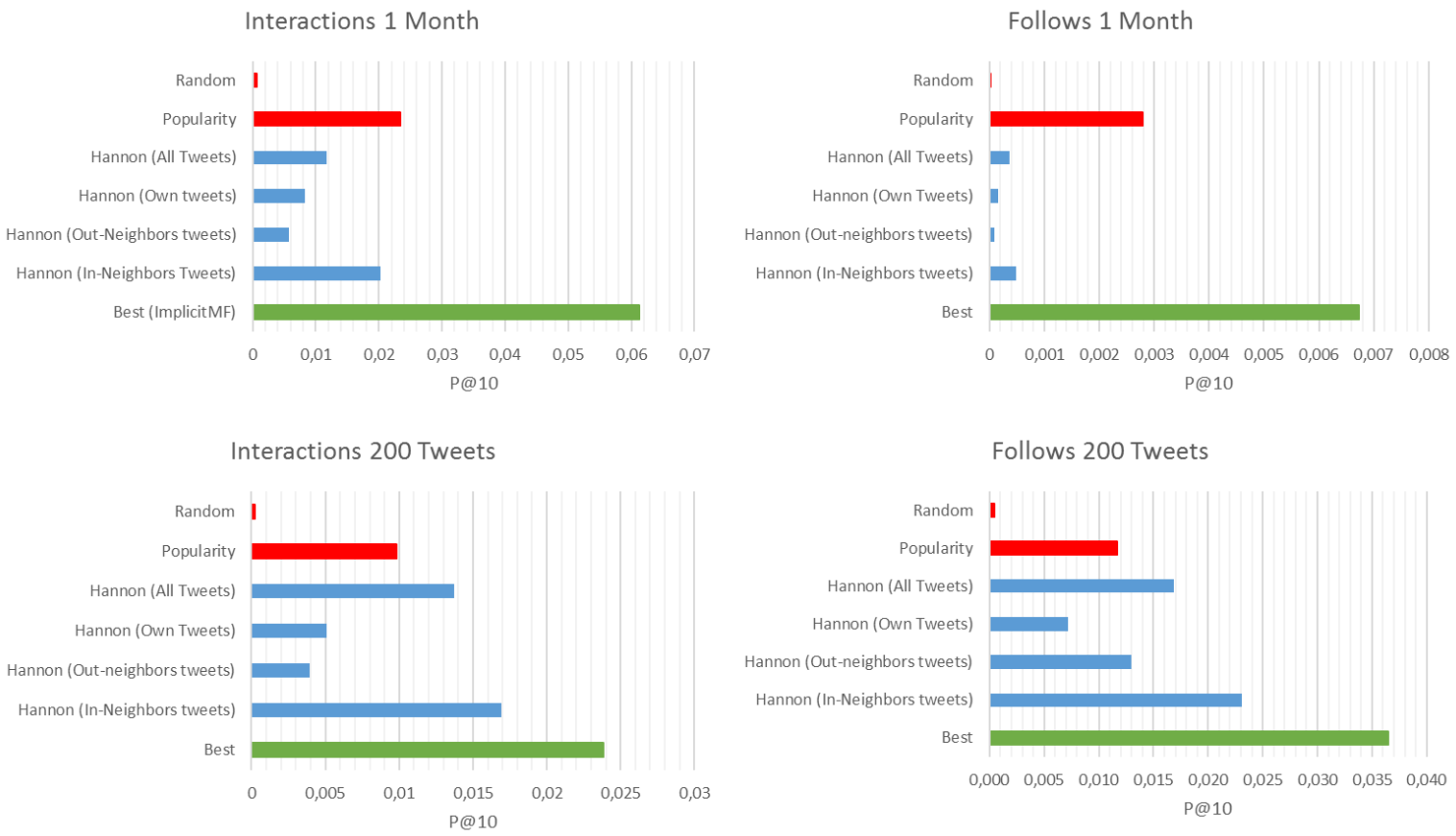


Figure 47. P@10 for the content-based algorithms

Finally, before exploring other evaluation perspectives, we will analyze the directionality of the edges in the recommendation approaches.

Most part of the link prediction approaches we have adapted to the contact recommendation problem were designed for their application over undirected networks. However, we run our experiments over different directed graphs, so the direction of the links is a variable to consider. Several algorithms, like PageRank, matrix factorization techniques or neighborhood-based collaborative filtering methods provide a direction for those edges in their definitions. However, other algorithms, like Adamic/Adar or BM25 admit several interpretations.

In our research, have adapted several some algorithms which generate recommendations taking the users intersection of the neighborhoods of the target user and the candidate user as a basis: Adamic/Adar, Resource Allocation (RA), all the variants of the Friend of a Friend algorithm (Jaccard, Salton,…), BIR, BM25, Extreme BM25 and Query Likelihood (QLJM). In those algorithms, for each one of the users, we can consider three different neighborhood: the in-neighborhood (the followers of the user), the out-neighborhood (the followees of the user), and the union of both (as if we consider that the edges are undirected). In total, there are nine possible configurations for those algorithms, which are shown in Figure 48. For those algorithms, we want to determine three aspects: first, the best option among the possible nine; second, the neighborhood that best describes the target user; third, the neigborhood that best describes the candidate users.

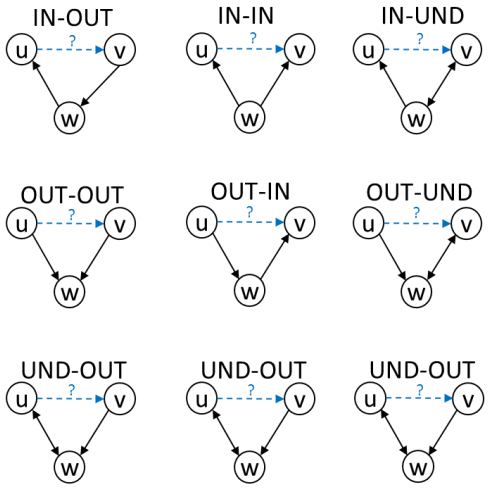


Figure 48. Different common neighborhoods options

To answer those questions, we have tried the different options for the different algorithms, and we have evaluated them in terms of P@10. Among those algorithms, Adamic/Adar and Resource Allocation pose a special case: the neighborhoods for the target and candidate users are not the only neighborhoods which must be chosen: we have to choose the neighborhoods for the users in the intersection first two. For those algorithms, we have executed all the possible options, and we have found that there is an outstanding option for the third neighborhood in all the graphs: choosing both the incident and adjacent neighbors of the users in the intersection.

In tables 15, 16, 17 and 18, we show the results for the different neighborhoods in four datasets. In those tables, each row represents the results for a single algorithm, and each column the selection of both neighborhoods, with the same nomenclature as the one used in Figure 48: the top header of the table represents the neighborhood selection

for the target user, and the second header represents the selection for the candidate user. In each row, the color of the cell represents how good the neighborhoods selections are: best options are shown in green, and the worst options in red. The best P@10 value for each algorithm is shown in each row of the table with white and bold numbers.

	IN			OUT			UND		
Algorithm	IN	OUT	UND	IN	OUT	UND	IN	OUT	UND
Adamic/Adar	0.0344	0.0113	0.0323	0.0461	0.0090	0.0186	**0.0506**	0.0107	0.0294
BIR	0.0345	0.0116	0.0324	0.0454	0.0093	0.0193	**0.0505**	0.0112	0.0299
BM25	0.0322	0.0069	0.0308	0.0414	0.0052	0.0220	**0.0466**	0.0061	0.0419
ExtremeBM25	0.0311	0.0055	0.0300	0.0400	0.0042	0.0253	**0.0452**	0.0047	0.0342
FOAF	0.0327	0.0105	0.0308	0.0437	0.0076	0.0158	**0.0484**	0.0092	0.0257
HDI	0.0107	0.0063	0.0092	0.0123	0.0056	0.0038	**0.0169**	0.0064	0.0079
HPI	0.0167	0.0062	**0.0213**	0.0098	0.0052	0.0081	0.0099	0.0048	0.0110
Jaccard	0.0113	0.0066	0.0109	0.0115	0.0062	0.0041	**0.0169**	0.0072	0.0091
LHN Index 1	0.0023	**0.0032**	0.0026	0.0023	0.0030	0.0013	0.0019	0.0029	0.0013
QLJM	0.0275	0.0101	0.0252	0.0375	0.0075	0.0092	**0.0435**	0.0091	0.0199
RA	0.0319	0.0111	0.0306	0.0394	0.0115	0.0239	**0.0451**	0.0132	0.0337
Salton	0.0107	0.0061	0.0112	0.0068	0.0063	0.0041	**0.0127**	0.0073	0.0098
Sorensen	0.0113	0.0066	0.0109	0.0115	0.0062	0.0041	**0.0169**	0.0072	0.0091
TF-IDF	0.0111	0.0067	0.0115	0.0083	0.0081	0.0058	**0.0139**	0.0093	0.0119

Table 15. P@10 comparison for the possible neighborhood selections of different recommendation algorithms (1 Month interactions)

We observe that three of the four networks show similar results: the interactions graph for both datasets, and the follows graph for the 200 Tweets datasets. In all those datasets, the best possible configuration is UND-IN, i.e., when we select the in-neighborhood of the candidate user, and choose all the neighbors of the target user. Two algorithms are an exception: Hub Promoted Index and the first Leicht-Holme-Newman Index. For the first algorithm, the best option is IN-UND and, for the second, IN-OUT is the most effective interpretation. In the remaining graph, although UND-IN is not the best option for most part of the algorithms, it is still a competitive neighborhood selection, and obtains similar results to the other effective options, like taking the all the neighbors of the candidate user and the in-neighbors of the target user (IN-UND), or taking all neighbors for both users (UND-UND).

Surprisingly, taking the followees of the target user and the followers of the candidate user, which seems the most obvious choice for neighborhoods does not achieve the better results in any dataset or algorithm. Although it is the second best option in the 1 Month interactions graph and the 200 Tweets follows graph, it even falls to the fourth or fifth best option for the remaining graph, far from our initial expectations.

	IN			OUT			UND		
Algorithm	IN	OUT	UND	IN	OUT	UND	IN	OUT	UND
Adamic/Adar	0.0014	0.0002	0.0012	0.0004	0.0001	0.0001	**0.0015**	0.0002	0.0011
BIR	0.0014	0.0002	0.0013	0.0004	0.0001	0.0001	**0.0015**	0.0002	0.0011
BM25	0.0045	0.0000	0.0045	0.0064	0.0001	0.0064	0.0067	0.0000	**0.0067**
ExtremeBM25	0.0045	0.0000	0.0045	0.0064	0.0001	0.0065	0.0067	0.0000	**0.0067**
FOAF	0.0013	0.0002	0.0012	0.0004	0.0001	0.0001	**0.0015**	0.0002	0.0010
HDI	0.0002	0.0001	**0.0003**	0.0002	0.0001	0.0001	0.0003	0.0001	0.0002
HPI	0.0009	0.0002	**0.0011**	0.0002	0.0001	0.0001	0.0009	0.0002	0.0009
Jaccard	0.0003	0.0001	**0.0004**	0.0002	0.0001	0.0001	0.0003	0.0001	0.0002
LHN Index 1	0.0001	0.0001	**0.0001**	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
QLJM	**0.0008**	0.0001	0.0007	0.0002	0.0001	0.0001	0.0007	0.0001	0.0005
RA	0.0015	0.0002	0.0013	0.0006	0.0001	0.0003	**0.0017**	0.0002	0.0013
Salton	0.0003	0.0001	**0.0004**	0.0001	0.0001	0.0001	0.0003	0.0001	0.0002
Sorensen	0.0003	0.0001	**0.0004**	0.0002	0.0001	0.0001	0.0003	0.0001	0.0002
TF-IDF	0.0003	0.0001	**0.0004**	0.0001	0.0001	0.0001	0.0003	0.0001	0.0003

Table 16. P@10 comparison for the possible neighborhood selections of different recommendation algorithms (1 Month follows)

	IN			OUT			UND		
Algorithm	IN	OUT	UND	IN	OUT	UND	IN	OUT	UND
Adamic/Adar	0.0452	0.0132	0.0411	0.0370	0.0128	0.0253	**0.0533**	0.0151	0.0428
BIR	0.0454	0.0129	0.0402	0.0364	0.0127	0.0277	**0.0534**	0.0148	0.0429
BM25	0.0458	0.0092	0.0423	0.0375	0.0084	0.0292	**0.0546**	0.0094	0.0461
ExtremeBM25	0.0457	0.0102	0.0391	0.0375	0.0094	0.0309	**0.0542**	0.0106	0.0446
FOAF	0.0420	0.0119	0.0384	0.0347	0.0107	0.0217	**0.0507**	0.0132	0.0391
HDI	0.0241	0.0128	0.0225	0.0177	0.0129	0.0100	**0.0312**	0.0145	0.0237
HPI	0.0064	0.0133	**0.0229**	0.0124	0.0080	0.0129	0.0044	0.0110	0.0159
Jaccard	0.0239	0.0143	0.0240	0.0164	0.0131	0.0107	**0.0299**	0.0157	0.0251
LHN Index 1	0.0029	**0.0123**	0.0089	0.0070	0.0084	0.0055	0.0027	0.0111	0.0075
QLJM	0.0396	0.0147	0.0381	0.0325	0.0127	0.0198	**0.0481**	0.0153	0.0381
RA	0.0416	0.0126	0.0390	0.0322	0.0134	0.0243	**0.0471**	0.0150	0.0407
Salton	0.0171	0.0150	0.0236	0.0124	0.0125	0.0113	0.0195	0.0162	**0.0238**
Sorensen	0.0239	0.0143	0.0240	0.0164	0.0131	0.0107	**0.0305**	0.0157	0.0251
TF-IDF	0.0195	0.0156	0.0248	0.0138	0.0142	0.0150	0.0221	0.0189	**0.0268**

Table 17. P@10 comparison for the possible neighborhood selections of different recommendation algorithms (200 Tweets interactions)

	IN			OUT			UND		
Algorithm	IN	OUT	UND	IN	OUT	UND	IN	OUT	UND
Adamic	0.0267	0.0163	0.0257	0.0315	0.0145	0.0198	**0.0328**	0.0148	0.0238
BIR	0.0268	0.0159	0.0256	0.0309	0.0150	0.0202	**0.0323**	0.0152	0.0240
BM25	0.0265	0.0144	0.0258	0.0311	0.0114	0.0226	**0.0326**	0.0120	0.0257
ExtremeBM25	0.0261	0.0129	0.0254	0.0306	0.0092	0.0237	**0.0321**	0.0098	0.0268
FOAF	0.0261	0.0158	0.0249	0.0301	0.0137	0.0184	**0.0316**	0.0140	0.0223
HDI	0.0160	0.0127	0.0141	0.0192	0.0123	0.0086	**0.0221**	0.0135	0.0135
HPI	0.0132	0.0126	**0.0192**	0.0120	0.0105	0.0123	0.0106	0.0099	0.0126
Jaccard	0.0173	0.0144	0.0163	0.0204	0.0137	0.0102	**0.0239**	0.0150	0.0151
LHN Index 1	0.0025	**0.0062**	0.0045	0.0026	0.0056	0.0025	0.0020	0.0058	0.0028
QLJM	0.0253	0.0169	0.0247	0.0309	0.0149	0.0169	**0.0328**	0.0155	0.0224
RA	0.0252	0.0151	0.0241	0.0311	0.0166	0.0229	**0.0322**	0.0165	0.0260
Salton	0.0170	0.0150	0.0180	0.0197	0.0144	0.0106	**0.0228**	0.0153	0.0156
Sorensen	0.0173	0.0144	0.0163	0.0204	0.0137	0.0102	**0.0239**	0.0150	0.0151
TF-IDF	0.0176	0.0156	0.0183	0.0206	0.0167	0.0129	**0.0233**	0.0172	0.0175

Table 18. P@10 comparison for the possible neighborhood selections of different recommendation algorithms (200 Tweets follows)

If we observe the neighborhoods for the target user and candidate user separately, all graphs show a similar behavior: the candidate users are not well represented by their outgoing neighbors: with the exception of LHN Index 1, the worst results are commonly obtained when that neighborhood is selected. Also, the best results are obtained when the selected neighbor for that user is the set of incident users. In the case of the target, we observe that the worst results are obtained when we use his out-neighbors, but it is not clear which of the remaining options (using the in-neighborhood or the whole set of neighbors) better describes the target user.

5.4.2 Other evaluation perspectives

Once we have finished studying the accuracy of the different contact recommendation algorithms in our comparative, we will discuss the most important results related to the novelty, diversity and structural diversity. Again, we have computed the different metrics only using the top 10 recommended users in the network.

Since we have proposed many metrics, a first step consists in the identification of highly correlated metrics. To select them, we have computed the Pearson correlation coefficient for each metric and graph, and we have selected those ones which obtain correlation coefficients greater than 0.8 (or smaller than -0.8) in all graphs. A relation of the highly (directly and inversely) correlated metrics is shown in Table 19.

One noticeable aspect of the correlations we observe between different metrics is that aspect-based diversity metrics, modularity and the number of weak ties do not depend on the chosen community detection algorithms (rankings for that metrics are almost the same). However, this does not apply for community Gini variants: in those cases, the relations between the metrics for different community detection algorithms is not as clear.

Metric	Positive correlation	Negative correlation
Modularity Louvain	Modularity Louvain, Modularity Leading Vector, Modularity Infomap	Profile Distance, Weak Ties Louvain, Weak Ties Leading Vector, Weak Ties Infomap
ERR-IA Louvain	ERR-IA Leading Vector, ERR-IA Infomap, S-Recall Louvain, S-Recall Leading Vector, S-Recall Infomap	
Comm. Pair-Gini Louvain	Comm. In-Gini Louvain	
Comm. Pair-Gini Leading Vector	Comm. In-Gini Leading Vector	
Comm. Pair-Gini Infomap	Comm. In-Gini Infomap	

Table 19. Highly correlated metrics

Another interesting aspect arises from the relation between profile distance (PD) and the number of weak ties / modularity: that metric, which measures the average distance between the target user and the top candidate users, obtains higher values when weak ties are recommended, showing that the distances between the profiles highly depend on the differences between communities.

In tables 20, 21, 22 and 23 we show a comparative for the most interesting algorithms and metrics for each graph. In that tables, each row represents an algorithm, and each column a different metric. A color gradient which goes from green to red is shown in every column. Green values represent the best values for the metric, while red ones represent the worst ones. For each metric, the best value in the comparative is highlighted with white and bold numbers, appart from the green background. The community related metrics (ERR-IA, modularity and community Gini) use the communities generated by the Louvain algorithm. The rest of the results, as we explained before, are shown in the second annex of this document.

Novelty and diversity metrics

First, we will compare the different algorithms in terms of recommender systems novelty and diversity metrics. Here, we differentiate three types of metrics: novelty, classic recommender systems diversity metrics and aspect-based diversity metrics. For the first two families (PC,PD,ILD and Gini coefficient metrics), we observe that recommending random users grants good values for all the metrics. Also, most accurate methods like BM25 or ImplicitMF obtain moderate (and even bad) values for all of them.

For the popularity complement metric (PC), the worst values are always achieved by the popularity-based recommendation. This is not surprising, since this metric represents the probability that a user has not previously interacted with the candidate user, and popular users represent the most widely known users in the network. We could say that PC measures how impopular are the recommended users. Random-walk based algorithms like PageRank, or the personalized versions of PageRank, SALSA or HITS also attain low values for this metric in all the datasets, showing that those algorithms may be similar to the popularity-based ones. Both Leicht-Holme-Newman indexes obtain the best values for the metric, due to the strong penalty they apply to popular items in the recommendation (as it can be seen in equations 3.9 and 3.22 for each index).

	Accuracy	Novelty	Diversity			Structural Diversity				
Algorithm	P@10	PC	ILD	Gini	ERR-IA	Clust. Coef.	Mod.	Comm. In-Gini	ASL	Rec. Dist
ImplicitMF (k=280,α=40;λ=150)	**0.0612**	0.9571	0.2012	0.0288	0.0721	0.0954	0.7084	0.3817	3.2663	2.34
UserBased kNN (k=120)	0.0598	0.9459	0.2365	0.0182	**0.0800**	0.0919	0.6720	0.3609	3.0509	2.38
Personalized SALSA (Auth.;r=0.99)	0.0577	0.9275	0.1872	0.0115	0.0748	0.0955	0.6927	0.3501	2.9905	2.31
Average Cosine Similarity	0.0554	0.9479	0.1027	0.0331	0.0729	0.1107	0.7149	0.3639	3.2765	2.30
Centroid Cosine Similarity	0.0541	0.9414	0.0993	0.0206	0.0693	0.1124	0.7041	0.3509	3.1922	2.31
Local Path Index (($\Gamma_{und}(u)$;β=0.1;l=3)	0.0530	0.9209	0.0701	0.0069	0.0698	0.1046	0.6992	0.3110	3.0792	2.34
Adamic/Adar($\Gamma_{und}(u)$;$\Gamma_{in}(v)$, $\Gamma_{und}(w)$)	0.0506	0.9585	0.2029	0.0564	0.0614	0.1173	0.7163	0.3804	3.1534	2.09
BIR ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$)	0.0505	0.9580	0.2125	0.0502	0.0606	0.1178	0.7121	0.3843	3.1636	2.08
BM25($\Gamma_{und}(u)$;$\Gamma_{in}(v)$;b=0.1;k=1.0)	0.0466	0.9640	0.1703	0.0549	0.0535	0.1395	0.7206	0.3935	3.3564	2.15
Commute Time	0.0239	0.9174	0.5350	0.0011	0.0253	0.0909	0.4262	0.3205	2.8526	2.69
Popularity	0.0234	0.8835	0.5247	0.0011	0.0300	0.0757	0.4253	0.2180	2.6718	2.73
Hitting Time	0.0224	0.9181	0.5598	0.0011	0.0237	0.0907	0.4205	0.3235	2.8488	2.70
Hannon ($\Gamma_{in}(u)$)	0.0203	0.9916	0.1173	0.2724	0.0240	0.1090	0.7245	0.4086	3.3959	2.36
PageRank (r=0.1)	0.0199	0.9171	0.7093	0.0011	0.0206	0.0850	**0.4116**	0.4089	2.7530	2.81
Personalized PageRank (r=0.4)	0.0181	0.9056	**0.7155**	0.0011	0.0191	0.0737	0.4263	0.3694	**2.6391**	2.85
Jaccard ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$)	0.0169	0.9963	0.2151	0.2122	0.0217	0.1100	0.7177	0.4306	3.3376	2.15
TF-IDF ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$)	0.0139	0.9966	0.2760	0.2528	0.0171	0.1026	0.6971	**0.4536**	3.2298	2.21
Katz ($\Gamma_{out}(u)$.;β=0.1)	0.0102	0.9737	0.3617	0.0142	0.0134	0.0886	0.4924	0.3629	3.1920	2.65
LHN Index 1 ($\Gamma_{in}(u)$;$\Gamma_{out}(v)$)	0.0032	0.9984	0.1667	0.0737	0.0037	0.0852	0.7310	0.3706	3.3706	2.53
LHN Index 2 (β=0.4)	0.0012	**0.9999**	0.0877	0.0857	0.0012	0.0906	0.6784	0.3972	3.4982	2.31
Random	0.0006	0.9980	0.6452	**0.8486**	0.0007	**0.0485**	0.4540	0.4020	2.9488	**3.54**

Table 20. Selection of some of the most interesting metrics and algorithms (Interactions 1 Month)

	Accuracy	Novelty	Diversity			Structural Diversity				
Algorithm	P@10	PC	ILD	Gini	ERR-IA	Clust. Coef.	Mod.	Comm. In-Gini	ASL	Rec. Dist
BM25($\Gamma_{und}(u)$;$\Gamma_{und}(v)$;b=1.0;k=1000)	**0.0067**	0.9469	0.51	0.56	0.0087	0.0782	0.5910	0.8807	3.2579	2.1133
PageRank (r=0.6)	0.0032	0.7439	0.5564	0.0018	0.0026	0.0743	0.5167	0.8846	3.0522	2.1779
Personalized PageRank (r=0.5)	0.0030	0.7411	0.5373	0.0018	0.0021	0.0741	0.5168	0.8447	3.0511	2.1717
Popularity	0.0028	0.7301	0.5039	0.0018	0.0024	0.0734	**0.5122**	0.8177	3.0416	2.1923
Local Path Index ($\Gamma_{und}(u)$;β=0.3;l=5)	0.0027	0.8328	0.1618	0.0046	0.0006	**0.0689**	0.5872	0.8794	3.0064	2.0863
Katz ($\Gamma_{out}(u)$.;β=0.9)	0.0020	0.9668	0.4823	0.0053	0.0019	0.0780	0.5374	0.7521	3.1286	2.6466
Adamic/Adar($\Gamma_{und}(u)$;$\Gamma_{in}(v)$, $\Gamma_{und}(w)$)	0.0015	0.8598	0.2457	0.0314	0.0012	0.0800	0.6259	0.8399	3.0247	2.0206
BIR ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$)	0.0015	0.8570	0.2504	0.0274	0.0012	0.0800	0.6251	0.8412	3.0301	2.0187
Personalized SALSA (Auth.;α=0.1)	0.0014	0.7431	0.5578	0.0027	0.0012	0.0749	0.5472	0.8580	3.0013	2.1212
ImplicitMF (k=30;λ=150.0;α=40.0)	0.0011	0.8904	0.2934	0.0494	0.0005	0.0827	0.6290	0.8544	2.9607	2.2313
User-Based kNN (k=70)	0.0008	0.8617	0.2339	0.0282	0.0006	0.0826	0.6327	0.8687	3.0237	2.0642
Commute Time	0.0007	0.9306	0.2001	0.0012	0.0003	0.0748	0.5386	0.6299	3.0828	2.2764
Hannon ($\Gamma_{in}(v)$)	0.0005	0.9778	0.2071	0.4240	0.0005	0.0959	0.6295	0.8780	3.1004	2.2493
TF-IDF ($\Gamma_{in}(u)$;$\Gamma_{und}(v)$)	0.0004	0.9863	0.2693	0.3478	0.0004	0.1005	0.6185	0.8693	3.0699	2.3489
Jaccard ($\Gamma_{in}(u)$;$\Gamma_{und}(v)$)	0.0004	0.9884	0.2476	0.2502	0.0006	0.0827	0.6245	0.8693	3.0808	2.3523
Hitting Time	0.0004	0.9369	0.1999	0.0012	0.0002	0.0741	0.5376	0.6012	3.0337	2.0382
Centroid Cosine Similarity	0.0004	0.8451	0.1244	0.0202	0.0004	0.0997	0.6199	0.8674	3.0813	2.3067
Average Cosine Similarity	0.0003	0.8639	0.1219	0.0331	0.0003	0.0868	0.6325	0.8629	3.0393	2.0381
LHN Index 1 ($\Gamma_{in}(u)$;$\Gamma_{und}(v)$)	0.0001	0.9956	0.3233	0.1125	0.0001	0.0807	0.6069	0.8839	3.0438	2.5130
LHN Index 2 (β=0.1)	0.0001	**0.9998**	0.1900	0.0380	0.0001	0.0799	0.6237	**0.9475**	3.1183	2.6580
Random	0.0000	0.9939	**0.6661**	**0.9713**	0.0000	0.0754	0.5387	0.8661	**2.7988**	**3.1553**

Table 21. Selection of some of the most interesting metrics and algorithms (Follows 1 Month)

	Accuracy	Novelty	Diversity			Structural Diversity				
Algorithm	P@10	PC	ILD	Gini	ERR-IA	Clust. Coef.	Mod.	Comm. In-Gini	ASL	Rec. Dist
BM25 ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$;b=0,3;k=1,0)	**0.0238**	0.9838	0.5400	0.1764	**0.0259**	0.0538	0.5864	0.5290	4.2038	2.4023
BIR ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$)	0.0233	0.9827	0.5514	0.1630	0.0252	0.0532	0.5791	0.5162	4.0103	2.3828
Adamic/Adar($\Gamma_{und}(u)$;$\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.0233	0.9829	0.5532	0.1806	0.0251	0.0548	0.5820	0.5229	3.9829	2.4675
ImplicitMF (k=300;λ=150;α=40)	0.0226	0.9848	0.5136	0.0595	0.0245	0.0433	0.5647	0.5629	3.9789	3.0264
Personalized SALSA (Auth,;α=0,99)	0.0209	0.9711	0.6177	0.0656	0.0222	0.0397	0.5279	0.5080	3.8700	2.5881
User-Based kNN (k=100)	0.0202	0.9755	0.5503	0.0528	0.0231	0.0394	0.5413	0.5114	3.8868	2.6749
Centroid Cosine Similarity	0.0179	0.9835	0.4240	0.1177	0.0217	0.0478	0.5593	0.4926	3.8846	2.7934
Hannon ($\Gamma_{in}(u)$)	0.0169	0.9918	0.4489	0.2590	0.0195	0.0781	0.4457	0.4337	4.1840	2.8471
Average Cosine Similarity	0.0162	0.9902	0.4646	0.1956	0.0207	0.0587	0.5739	0.5601	3.9123	2.9208
Jaccard ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$)	0.0133	0.9981	0.5184	0.4446	0.0154	0.0928	0.5877	0.5480	3.9638	2.9692
TF-IDF ($\Gamma_{und}(u)$;$\Gamma_{und}(v)$)	0.0117	0.9988	0.4788	0.5088	0.0137	0.0993	0.6150	0.5924	3.9469	3.4200
Popularity	0.0098	0.9387	0.6319	0.0010	0.0067	0.0178	**0.3259**	0.3282	3.9679	3.0832
PageRank (r=0,8)	0.0089	0.9433	0.7191	0.0010	0.0063	**0.0169**	0.3266	0.4000	3.8745	3.1875
Personalized PageRank (r=0,4)	0.0068	0.9601	0.7745	0.0009	0.0052	0.0205	0.3308	0.3729	4.2050	3.0804
Local Path Index ($\Gamma_{in}(u)$;β=0,1;l=3)	0.0057	0.9967	0.4882	0.1473	0.0067	0.0942	0.5609	0.5337	3.4705	3.2857
Commute Time	0.0054	0.9699	0.9277	0.0010	0.0039	0.0221	0.3424	0.3903	4.5067	3.1353
LHN Index 1 ($\Gamma_{out}(u)$;$\Gamma_{in}(v)$)	0.0054	0.9989	0.6513	0.2291	0.0058	0.0962	0.5339	0.5581	3.8145	3.4972
Hitting Time	0.0051	0.9702	0.9312	0.0009	0.0036	0.0217	0.3379	0.3857	4.5113	3.1515
LHN Index 2 (β=0,1)	0.0034	**0.9998**	0.5967	0.1324	0.0031	0.0582	0.5495	**0.6211**	4.2094	3.8137
Katz ($\Gamma_{out}(u)$,;β=0,1)	0.0032	0.9862	0.5565	0.0099	0.0035	0.0203	0.3532	0.4114	3.8743	3.3607
Random	0.0002	0.9985	**0.9512**	**0.8278**	0.0003	0.0205	0.3359	0.5757	**3.3479**	**4.7038**

Table 22. Selection of some of the most interesting metrics and algorithms (Interactions 200 Tweets)

	Accuracy	Novelty	Diversity			Structural Diversity				
Algorithm	P@10	PC	ILD	Gini	ERR-IA	Clust. Coef.	Mod.	Comm. In-Gini	ASL	Rec. Dist
ImplicitMF (k=300;α=40;λ=150)	**0.0365**	0.9694	0.3746	0.1174	**0.0371**	0.0951	0.5694	0.7765	3.2160	2.1409
User-based kNN (k=40)	0.0343	0.9599	0.3053	0.0860	0.0363	0.0938	0.5694	0.7637	3.1885	2.1363
Average Cosine Similarity	0.0338	0.9624	0.2100	0.1126	0.0359	0.0987	0.5694	0.7547	3.2226	2.0919
Adamic/Adar($\Gamma_{und}(u)$;$\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.0328	0.9516	0.3253	0.0948	0.0326	0.0911	0.5694	0.7469	3.1913	2.0620
BM25 ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$;b=0.1;k=1)	0.0326	0.9524	0.3223	0.0960	0.0323	0.0923	0.5694	0.7515	3.2389	2.0543
BIR ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$)	0.0323	0.9511	0.3193	0.0872	0.0323	0.0906	0.5694	0.7460	3.1956	2.0519
Centroid Cosine Similarity	0.0316	0.9525	0.2024	0.0685	0.0325	0.0921	0.5694	0.7334	3.2414	2.2067
Personalized SALSA (Auth;α=0.99)	0.0304	0.9332	0.3959	0.0380	0.0295	0.0860	0.5616	0.7266	3.1826	2.0768
Local Path Index ($\Gamma_{out}(u)$;l=2;β=0.2)	0.0296	0.9555	0.3370	0.0733	0.0302	0.0906	0.5694	0.7597	3.2309	2.0111
Jaccard ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$))	0.0239	0.9918	0.3351	0.4692	0.0254	0.1121	0.5694	0.7955	3.2229	2.1339
TF-IDF ($\Gamma_{und}(u)$;$\Gamma_{in}(v)$))	0.0233	0.9925	0.3950	**0.5251**	0.0253	0.1135	0.5694	**0.8030**	3.1636	2.1897
Hannon ($\Gamma_{in}(u)$))	0.0231	0.9805	0.3257	0.3034	0.0239	0.1050	0.5694	0.7948	3.2414	2.2067
Popularity	0.0117	0.8772	0.5831	0.0012	0.0058	**0.0693**	0.4572	0.7095	3.1193	2.3435
Personalized PageRank (r=0,8)	0.0091	0.8984	0.7836	0.0011	0.0044	0.0727	**0.4504**	0.6910	3.1467	2.3339
PageRank (r=0.8)	0.0091	0.8984	**0.7836**	0.0011	0.0044	0.0727	0.4504	0.6910	3.1467	2.3338
LHN Index 1 ($\Gamma_{in}(u)$;$\Gamma_{out}(v)$)	0.0062	0.9969	0.6206	0.2009	0.0055	0.0968	0.5406	0.7921	3.1693	2.6365
Katz ($\Gamma_{out}(u)$;β=0.4)	0.0029	0.9730	0.7290	0.0067	0.0016	0.0805	0.4579	0.7014	3.3654	2.7148
LHN Index 2 (β=0.2)	0.0028	**0.9996**	0.5516	0.1227	0.0026	0.0898	0.5597	0.8021	3.2397	2.8077
Hitting Time	0.0027	0.9640	0.4010	0.0010	0.0012	0.0798	0.4597	0.5932	3.3136	2.4545
Commute Time	0.0027	0.9643	0.5216	0.0010	0.0013	0.0794	0.4597	0.5931	3.3096	2.4772
Random	0.0005	0.9953	**0.8581**	**0.8793**	0.0003	0.0758	0.4581	0.7913	**2.8904**	**3.3700**

Table 23. Selection of some of the most interesting metrics and algorithms (Follows 200 Tweets)

This is supported by normalized common neighbors methods like Jaccard or Salton, which penalize the popularity of both the target and the candidate users, since they also reach high positions in the algorithm ranking for PC.

Among the most accurate algorithms, we observe that the different BM25 variants recommend users more equally (higher values for the Gini coefficient), ImplicitMF differs most from popularity (higher values for PC), and Personalized SALSA provides, in general, more novel and diverse recommendations for the users (higher values for PD and ILD)

Structural diversity metrics

Finally, we will study the effects of the different algorithms in the structural diversity of the networks. In general, we observe that recommending random users grants a great structural diversity in terms of most part of the studied metrics, like modularity, clustering coefficient or distances.

One of our focuses on the study of the structural diversity has been the notion of weak link (defined as a link between two different communities). As we have seen in Table 19, modularity metrics for the different community detection algorithms are highly correlated, so, from now on, we will focus on Louvain communities. We observe that not personalized algorithms like PageRank or Popularity, as well as algorithms with low Gini coefficient values like Commute Time, Hitting Time or PMF variants recommend a number of weak ties higher than the rest, and therefore modularity values are low. The reason for that is simple: since all the recommendation rankings are very similar, the modularity is determined by the number of users in different communities than the recommended ones: if this number is high, then, the number of weak links would be high, and thus the values of the modularity would be low. However, since all links are distributed among the same communities, we observe that community in-Gini values for those algorithms are very low.

Algorithms like BM25, Adamic or Jaccard usually recommend less weak ties to the different target users: attending to their formulation, these algorithms close triangles (depending on the configuration of the algorithm, one of the 9 possible triangles shown in Figure 48). The endpoints of redundant edges are more likely to belong to the same communities, and, since these nodes are the ones recommended by those algorithms, that may explain their high modularity values. From that family of algorithms, TF-IDF, Jaccard, Salton or Sørensen attain high values for the community in-Gini metric (showing that, altough less weak ties are recommended, they are better distributed among communities). From the most effective approaches, the personalized version of the SALSA algorithm proposed by Twitter obtains the better modularity values, but the values are still far from optimal. BM25 and variants, and implicitMF, in general, obtain big values for the modularity and medium values for the community in-Gini metric.

Studying the community in-Gini metric for other community detection algorithms, we observe that the metric variant which uses Leading communities algorithms show very different values than the ones which use Louvain and Infomap communities: sometimes, even popularity-based recommendation achieves good values for the metric. Observing the data in tables 4, 5, 9 and 10, we see that Leading Vector usually detects a smaller number of different communities than the other two algorithms, which may explain those differences. In general, although community in-Gini for Louvain communities and community Gini for Infomap communities are not highly correlated,

they are indeed positively correlated, and both best and worst algorithms for Infomap communities are usually the same as the ones for Louvain.

In our investigation, we have also considered the notion of local bridge proposed by Granovetter (1973): the number of links in the network with embeddedness equal to zero. Appart from random recommendation, which maximizes the number of these links, two algorithms obtain good values in all four graphs: both LHN indexes. Worst values are obtained by methods based in common neighbors like Salton, Jaccard or BIR, which tipically use all neighbors of the target user and the incident neighbors of the candidate user to generate the recommendations in their best configurations.

Observing the embeddedness of the network for the best variants of each algorithm we observe that good values are obtained by algorithms like popularity-based recommendation algorithms, PageRank, Personalized PageRank, etc. From the algorithms which obtain the worst values for this metric, the most notorious are the variants of FOAF. As we stated before, those algorithms close triangles in the network, so it is not surprise that this metric obtains higher values (and therefore less diverse values) for that set of algorithms. Another algorithm which obtains bad edge embeddedness values is the Hannon content-based approach. From the most effective recommenders, we have observed that personalized SALSA obtains small values for this metric. ImplicitMF and BM25, however, still obtain high embeddedness values in all graphs.

Studying the values of clustering coefficient, we observe that algorithms like Katz or the first LHN Index obtain good values in all datasets. Worst values in three of the four datasets (excepting the interactions graph of the 1 month dataset) are obtained by popularity-based recommendation and similar methods, like PageRank. It is noticeable that ImplicitMF, as a representant of highly accurate methods, obtains good values for this metric in all four graphs.

Metrics like ASL, average eccentricity or the diameter of the network do not provide conclusive results about what algorithms are more diverse in terms of distances: in this case, best algorithms highly depend on the studied dataset. However, there are some algorithms which work bad for all the different metrics in all the different datasets: Hannon recommendation algorithm, BM25 and Extreme BM25 algorithms.

Finally, if we study the recommendation distance of the algorithms, we find that algorithms like Katz or LHN Index 2 obtain good values, and recommend contacts far from the target user. On the other side of the ranking, Adamic/Adar, BIR, BM25 and Extreme BM25 usually recommend very close users (tipically, at distance 2).

5.4.3 Conclusions

In conclusion, we have identified several algorithms which obtain highly competitive values in the different studied graphs. From the whole set, two algorithms obtain the higest values: BM25 variants and matrix factorization for implicit feedback (ImplicitMF). As we stated, other algorithms like the personalized SALSA algorithm used in the Twitter Who-To-Follow system, Adamic-Adar algorithm or user-based kNN also provide accurate recommendations. The worst recommendations are always provided by both PMF variants, and random recommendation.

However, the most accurate methods do not work so well in terms of novelty, diversity and structural diversity, usually obtaining medium or bad values for the different studied metrics. In all graphs, random recommendation trivially outstands in

most of the metrics. Aspect-based diversity metrics make an exception to this, since they are highly correlated to the accuracy metrics.

In terms of novelty and diversity metrics, among the most accurate algorithms, we have found that the different BM25 variants recommend users more equally, ImplicitMF differs most from popularity, and Personalized SALSA provides, in general, more novel and diverse recommendations for the users.

In terms of structural diversity, personalized SALSA outstands in terms of modularity and clustering coefficient. ImplicitMF obtains very high values (and therefore bad values) for modularity, but it provides a good balance between precision and community Gini metrics. ImplicitMF, as well as user-based kNN are also good in terms of distance metrics.

Finally, although we have found many differences between different graphs, we have not been able to differentiate follows and interactions graphs: in terms of accuracy, three of the four studied graphs showed similar behaviors, and, for most part of the metrics studied for other perspectives, we observed similar properties for the different algorithms in all the datasets.

6. Information diffusion

Passing and catching information is of the foremost uses of (online and offline) social networks. It is known that the information diffusion is highly influenced by the structure of the social network, and several models and theories explaining that influence have been documented (Granovetter 1973, Doerr et al. 2011, De Meo et al. 2014). In the previous chapters, we studied how recommender systems affect the evolution of networks, so it seems natural to combine both lines of thought to study how recommender systems influence the flow of information that runs through social networks. In this chapter we study the indirect effect of recommendations on different properties of information diffusion, like the speed or the information diversity, via the direct effects of the recommendation on the structure of the network. This is illustrated in Figure 49.

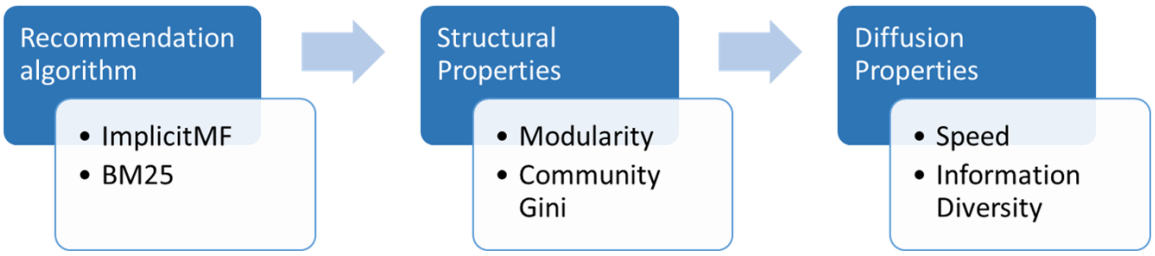


Figure 49. Effects of contact recommendation

We study these effects using empirical simulations, where the input are the different recommendation algorithms and the output are the properties of the diffusion. We aim to show which structural properties of the recommendation correspond with structural properties of the graph. Apart from the own interest of this study, we want to study the sense of some of the metrics we have defined and used in previous chapters, like community Gini.

6.1 Research Questions

The study that follows and the reported experiments address the following research questions:

RQ1. Do contact recommender systems impact the properties of the information flow through the network?

RQ2. Does the enhancement of the structural diversity have an effect on the properties of the flow of information through the network?

RQ3. Is it useful to recommend weak ties?

RQ4. Are the answers to the above questions sensitive to the communication protocol between users in the social network platform?

6.2 Structural Diversity Enhancement

One of the main objectives of the present research consists in the study of how the structural diversity properties of a certain contact recommendation algorithm affect the diversity of the information that flows through the recommendation network. A way to measure the impact of a certain property on the information diversity consists in enhancing the different structural measures of a generated recommendation, and observe the effects of that enhancement over the flow of information.

In recent years, many methods have been documented for the improvement of a property of a recommender system (mainly, related to novelty and diversity). A common approach to enhance the novelty or the diversity of a recommendation is on the diversification or re-ranking of the results returned by an initial recommender system (Castells et al. 2015). Given an initial ranking for a recommendation, a subset of the candidate users selected by the algorithm are reordered to improve the diversity of the system.

It is also common to optimize not only the diversity of a system, but the relevance of the recommended users. In that case, the solution of a re-ranking problem is an ordered set of users, S, which maximizes the following equation:

$$\arg\max_{S \subset R(u)} (1-\lambda) \sum_{v \in S} f_u^{rel}(v) + \lambda div(S) \tag{6.1}$$

where $f_u^{rel}(v)$ is a function that depends on the original score for user v, $div(R)$ measures the value of the diversity of the ranking and $\lambda \in [0,1]$ represents the trade-off between relevance and diversity. Obtaining an optimal solution for this problem is not trivial, so greedy approaches are commonly used.

A simple greedy approach is the method known in Information Retrieval as maximum marginal relevance (MMR, Carbonell et al. 1998). That method iteratively adds to the diversified ranking the candidate user which maximizes a certain objective function, $g_u(v)$. If we define S^j as the ranking in the j-th iteration of the algorithm, then, the objective function is

$$g_u(v) = (1-\lambda) f_u^{rel}(v) + \lambda\, div\left(S^{j-1} \cup \{v\}\right) \tag{6.2}$$

The values of $f_u^{rel}(v)$ and $div\left(S^{j-1} \cup \{v\}\right)$ must be comparable, so it is necessary to normalize both scores before computing the objective function. One of the most used methods is rank-sim (Lee 1997), which normalizes the scores in the following way:

$$\text{rank-sim}\big(h(v)\big) = \frac{h(v) - \max\limits_{w \in \mathcal{U}} h(w)}{\max\limits_{w \in \mathcal{U}} h(w) - \min\limits_{w \in \mathcal{U}} h(w)} \tag{6.3}$$

The greedy algorithm we have just explained can be used to enhance properties of the recommendation algorithms, like ILD or the number of weak ties which are globally defined (or, at least, can be globally defined) as an aggregation or average of the values obtained for a single user. In those cases, the enhancement of the metric for a single user directly produces an enhancement of the global metric. This type of rerankers have several advantages: any new recommendation produced by the system can be immediately reranked, producing an enhancement of the diversity of the system, and several recommendations can be reordered concurrently.

6.2.1 Enhancement of global properties

A problem appears when the global property which has to be enhanced cannot be expressed as an aggregation or average of individual values. As it is illustrated in the example on Figure 50, greedy approaches like the previous one might be useless for enhancing that particular property.

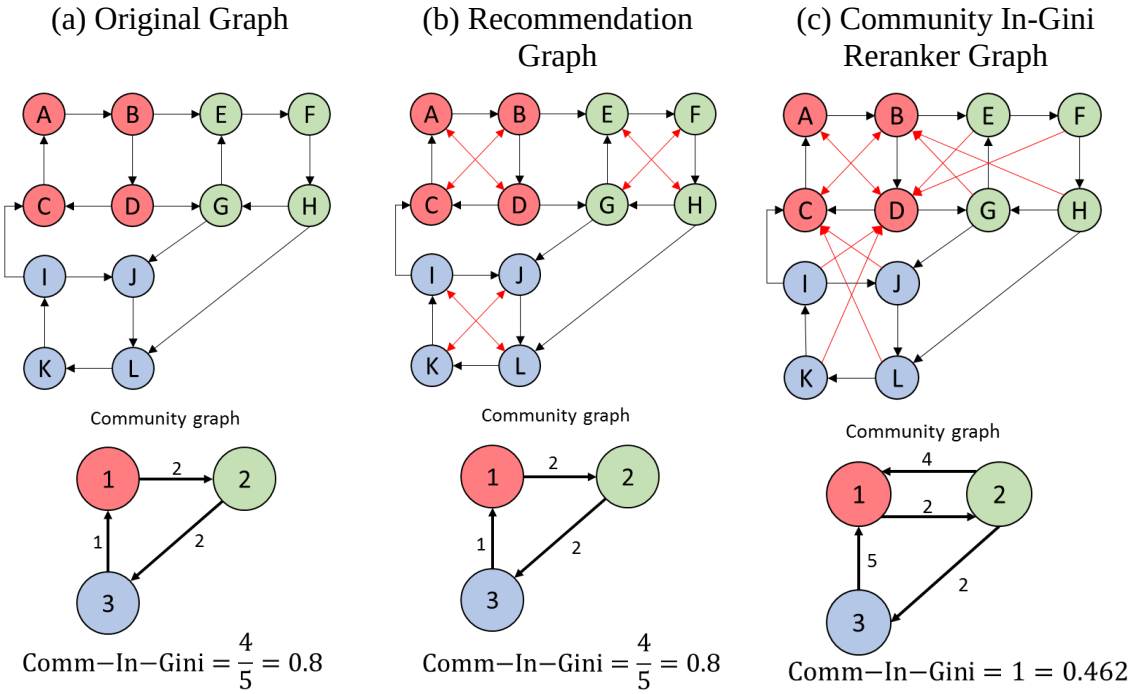


Figure 50. Independent top-1 re-ranking for enhancing community in-Gini example. Black arrows represent the existing links and red arrows the recommendation links.

In that example, we try to optimize the community in-Gini of the network, and each user is recommended a single link. When we rerank the recommendation for optimizing that metrics, all nodes in communities 2 and 3 would be recommended nodes in the first community (a single user from those communities to the first one would have the same number of incoming edges, so community in-Gini would reach the maximum value). The recommendations for community 1 would stay the same, since recommending a new weak tie would decrease the value of the metric. The total effect of those recommendations would be a total of 8 new links towards community 1, and none to the other ones. As a result, the in-degree of those communities becomes more unbalanced, and the community in-Gini value for the new graph is diminished.

As a result, it is necessary to define new approaches which allow enhancing global properties of the recommender systems. Although there is not much work in this line, this is not a new problem: several methods have already been proposed for increasing a global diversity metric known as the aggregate diversity. For example, Adomavicius & Kwon (2012) proposed a method for enhancing this metric using a combination of the score produced by the recommendation algorithm and another score based on the inverse popularity of the items.

In the present work, we propose a novel approach to the reranking problem that allows a direct joint optimization of both relevance of the recommended contacts and global properties of the recommender systems, like the structural diversity ones. In this method, users are not treated independently for the enhancement of the metric: the new

rankings depend on the recommendations produced for the rest of users in the network. In our method, we try to find a set $\mathcal{S} = \{S(u)|u \in \mathcal{U}\}$ of ordered rankings which optimize the following function:

$$\arg\max_{\substack{\mathcal{S} \\ S(u)\subset\mathcal{R}(u)}} (1-\lambda)\left[\sum_{u\in\mathcal{U}}\sum_{v\in S(u)} f_u^{rel}(v)\right] + \lambda\, div(\mathcal{S}) \tag{6.4}$$

where $div(\mathcal{S})$ represents the value of the global property we want to enhance.

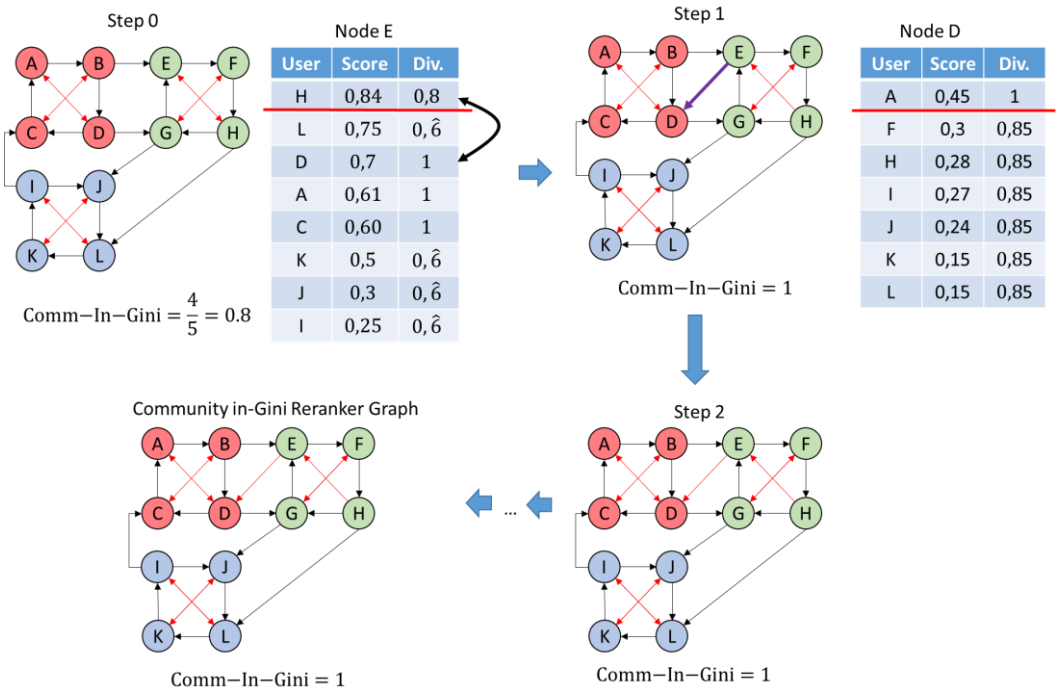


Figure 51. Community in-Gini global top-1 re-ranker example (λ=1)

The greedy algorithm we propose re-ranks each user ranking separately, but considering the rankings for the rest of users. Each iteration of the algorithm, an user is randomly selected (from the set of users whose recommendations had not been yet re-ranked). Then, the re-ranking of the generated recommendation for that user is done as follows:

First of all, we take the original recommendation ranking. We fix the top k elements of that ranking. Then, iteratively, we try to swap each element in the top with another candidate user in the ranking. For doing that, we select the recommended user that obtains the higher value for the following function:

$$g_u(v) = (1-\lambda) f_u^{rel}(v) + \lambda \cdot div\left(\tilde{\mathcal{S}}(u,i,v)\right) \tag{6.5}$$

where $\tilde{\mathcal{S}}(u,i,v)$ represents the full set of rankings for all users at cut-off k (we only take the first k elements of each ranking), with the i-th candidate user of the ranking for user u swapped for the user v. If the score obtained by that candidate user is greater than the score for the i-th node of the ranking, both nodes are definitely swapped in the recommendation ranking. If not, the ranking remains the same. An example of this algorithm is shown in Figure 51.

The values of both recommendation algorithms scores and diversity scores have to be comparable for this algorithm too, in order to correctly re-rank the recommendations, so it is necessary to apply over both values a normalization function like rank-sim.

6.3 Metrics

In our experiments, we measure two different properties of the information diffusion: the diffusion speed and the information diversity. In order to evaluate and compare the effects that recommendation algorithms produce on the information flow, we have defined a series of metrics, which are defined in this section. These metrics depend on two different properties of the dataset: the communities and the hashtags defined in the different published tweets.

6.3.1 Speed metrics

The diffusion speed is defined as the number of nodes that receive a certain piece of information over time. Several studies have addressed this perspective for several graph properties and diffusion models. For example, De Meo et al. (2012), using the Independent Cascade Model (Kempe et al. 2003), showed that the diffusion speed in the network was reduced if weak links were removed from the network, and Doerr et al. (2011) demonstrated that preferential attachment networks (Barabási & Albert 1999) spread rumors faster than the classical random model proposed by Erdös & Rényi (1959) thanks to the existence of low degree nodes using a variant of the Push-Pull Model.

In our experiments, we use a slightly different, although equivalent, definition: the number of different information pieces which have arrived to a certain user in a given time. Since we are using Twitter data, each information piece corresponds to a single tweet. We define the speed of the information flow as:

$$\text{speed}(T) = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\sum_{t=1}^{T}|\text{receivedTweets}(u,t)| \tag{6.6}$$

where $\text{receivedTweets}(u,t)$ represents the number of different tweets that have arrived to a certain user u at time t. It is important to show that the previous formula is equivalent to the classical speed definition (except for a constant factor). If we define $\mathcal{I}$ as the set of tweets we want to propagate, then:

$$\begin{aligned}\text{speed}(T) &= \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\sum_{t=1}^{T}|\text{receivedTweets}(u,t)| = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\sum_{t=1}^{T}\sum_{i\in\mathcal{I}}\mathbb{I}_{\text{receivedTweets}(u,t)}(i)\\ &= \frac{1}{|\mathcal{U}|}\sum_{t=1}^{T}\sum_{i\in\mathcal{I}}\sum_{u\in\mathcal{U}}\mathbb{I}_{\text{receivingUsers}(i,t)}(u)\\ &= \frac{1}{|\mathcal{U}|}\sum_{i\in\mathcal{I}}\sum_{t=1}^{T}|\text{receivingUsers}(i,t)| \propto \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\sum_{t=1}^{T}|\text{receivingUsers}(i,t)|\end{aligned}$$

where $\text{receivingUsers}(i,t)$ is the number of users which have received the information piece (tweet) i at time t.

6.3.2 Diversity metrics

The diversity of the information measures how different are the information pieces received by the users. A diverse flow of information might be useful for the people in the social network, since it can cause the discovery of new contents or different points of view for the known ones. The diversity of the information has been largely discused in the media in recent times[15], in relation to the so-called filter bubble effect (Pariser 2011): that effect occurs when a search engine or a recommender system isolates the information that a user receives, showing him only information he likes, or, at least, agrees with. Since a great number of recommender systems are personalized (including contact recommendation ones), they can potentially suffer from this effect. Several studies have evaluated the filter bubble effect caused by recommender systems (Nguyen et al. 2014, Bakshi et al. 2015). Learning how the structural diversity properties of a recommendation algorithm could enhance the diversity of the information that flows through the graph and thus, paliate this effect.

It is not easy to define what a diverse flow of information means. In our experiments, we will define the diversity of the flow in terms of the hashtags contained in the tweets. We have defined several metrics:

Hashtag Recall

This metric computes the average number of hashtags that a user in the network has received. It is computed as:

$$\text{HT-recall}(T) = \frac{1}{|\mathcal{U}||\mathcal{H}|}\sum_{u\in U}|\mathcal{H}_u(T)| \tag{6.7}$$

where $\mathcal{H}_u(T)$ is the set of hashtags that user u has received from the start of the difussion process until time T.

Since this measure could be influenced by the number tweets which have been propagated in a single iteration, an alternative metric, known as **Normalized Hashtag Recall** (HT-nRecall) is defined:

$$\text{HT-nRecall}(T) = \frac{1}{|\mathcal{U}||\mathcal{H}|}\sum_{u\in\mathcal{U}}\frac{|\mathcal{H}_u(T)|}{|\mathcal{I}_u(T)|} \tag{6.8}$$

where

$$\mathcal{I}_u(T) = \bigcup_{t=1}^{T}\text{receivedTweets}(u,t) \tag{6.9}$$

Hashtag Global Gini

This metric measures how equally hashtags have been propagated through the network. The number of times a user has been propagated is computed as the sum of all the different tweets received by users since the beginning of the diffusion:

[15] http://newsroom.fb.com/news/2015/05/news-feed-fyi-exposure-to-diverse-information-on-facebook/ (Accessed 15/12/2016)
http://www.recode.net/2016/12/1/13800270/facebook-filter-bubble-fix-technology-yann-lecun (Accessed 15/12/2016)

$$\text{HT-prop}(h,T) = \sum_{u \in \mathcal{U}} |\{i \in \mathcal{I}_u(T) | h_k \in \mathcal{H}_i\}| \tag{6.10}$$

where $\mathcal{H}_i$ represents the set of hashtags that tweet i contains. Then, Hashtag Global Gini is defined as:

$$\text{HT-GlobalGini(T)} = 1 - \frac{1}{|\mathcal{H}| - 1} \sum_{k}^{|\mathcal{H}|} (2k - N - 1) p(h_k | s, T) \tag{6.11}$$

where h_k is the k-th least propagated hashtag, and

$$p(h_k | s, T) = \frac{\text{HT-prop}(h,T)}{\sum_{h' \in \mathcal{H}} \text{HT-prop}(h',T)} \tag{6.12}$$

It can reach values in the interval [0,1]. It reaches value 1 when all hashtags have been propagated the same number of times, and 0 when only a hashtag has been propagated through the network.

Hashtag User Global Gini

Hashtag Global User Gini determines how balanced is the number of users which have received each one of the hashtags. This metric is defined as:

$$\text{HT-GlobalUserGini}(T) = 1 - \frac{1}{|\mathcal{H}| - 1} \sum_{k}^{|\mathcal{H}|} (2k - N - 1) p(h_k | s, T) \tag{6.13}$$

where h_k is the hashtag which has reached the k-th smallest set of users and

$$p(h_k | s, T) = \frac{|\{u \in \mathcal{U} | h_k \in \mathcal{H}_u(T)\}|}{\sum_j |\{u \in \mathcal{U} | h_j \in \mathcal{H}_u(T)\}|} \tag{6.14}$$

where $\mathcal{H}_u(t)$ represents the set of hashtags the user u has received at time t. This metric takes values in [0,1]. It reaches value 1 when all hashtags reach the same number of users, and 0 when only a single hashtag reaches users.

6.4 Experimental configuration

To answer the previously proposed research questions, we have designed a series of experiments oriented to check the effects of recommender systems on the flow of information through the network. In this section, we detail the design of those experiments, as well as several configuration considerations.

6.4.1 Simulation

In our experiments, we will simulate the diffusion of information through different graphs. Each one of the graphs we use in our experiments represents a different recommendation algorithm (re-ranked or not). Those graphs consist of the training graph we used for the experiments in the previous chapter, with the addition of the top k elements of each recommendation generated by the algorithm we want to represent.

For simplifying, our simulation models consider that time is discrete: the diffusion of information is made iteratively, and each iteration is considered a time step. The communication between different users only occurs in those time steps. The information that those users will spread through the network consists of a set of tweets obtained from the Twitter API. More details on those tweets will be provided in the next section.

In each iteration, the different users will select two different tweets to propagate: a tweet generated by themselves (a tweet they actually published on Twitter), and a tweet to from the ones they have received. Those pairs of tweets form the whole information that spreads through a network in a single iteration. Then, they will decide which users will receive that tweets, and the information will be sent. The receivers will automatically discard those tweets which they have already propagated (since users do not usually retweet something twice).

Appart from the selection of tweets and the receiver users, an additional factor has to be considered in our simulations: the number of time steps a user is likely to propagate a tweet after it is received: usually, the probability that a tweet is propagated after its reception decreases over time (it appears in lower positions of the timeline, making it more difficult to notice for the users). To manage that, after the couple of tweets are selected, and before receiving new ones, each user checks the different received tweets which has not propagated in order to determine if they could be retweeted in future iterations. If not, they are discarded.

The decision making for the selection of the receiver users and the expiration of the received tweets depends, in our case, of the communication protocol. The communication protocol or communication model plays a major role in the way the information is propagated through the nodes in the network. Several models have been proposed in the literature, like the Independent Cascade Model (Goldenberg et al. 2001), the Linear Threshold Model (Kempe et al. 2003) or the Networked Coordination Game (Eisley et al. 2010).

In our experiments, we use two different protocols, which we describe next: the Twitter protocol and the Push-Pull protocol. We have selected those algorithms since they may represent two of the main ways of spreading information in Twitter: the first one models the spread of information through the users' timeline, and the second one models the communication using direct private messages between pairs of users.

Twitter Protocol

As we have stated before, this protocol models the way the information is usually spread on Twitter. In that social network, the information published by a user in the form of tweets, or the forwarded information in form of retweets are shown to all the set of followers through the Twitter timeline. A user can see all the tweets published by the users he is following.

We have adapted this into a diffusion model. In each iteration, each users selects the full set of followers to spread the information. Figure 52 shows the diffusion links. Once the tweets for propagation have been selected, all the received tweets which have not been propagated expire, and they will not be propagated to other users. The diffusion ends when users do not receive new pieces of information.

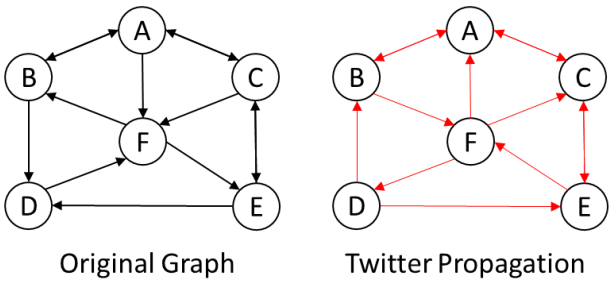


Figure 52. Twitter model diffusion links example

Push-Pull Protocol

The push-pull protocol was first defined by Demers et al. (1987). For each node in the network, these algorithms selects randomly a node. Both nodes (the selector and the selected) exchange them the information they have. Then, the selector node is not able to select the previous node until some time has passed. Doerr et al. (2011) used a particular version of this protocol to show that a single piece of information spreads faster in a preferential attachment graph than in Erdös-Rényi random graph. In that version, the algorithm selects, for each user, a random neighbor from the ones which were not selected in the previous iteration.

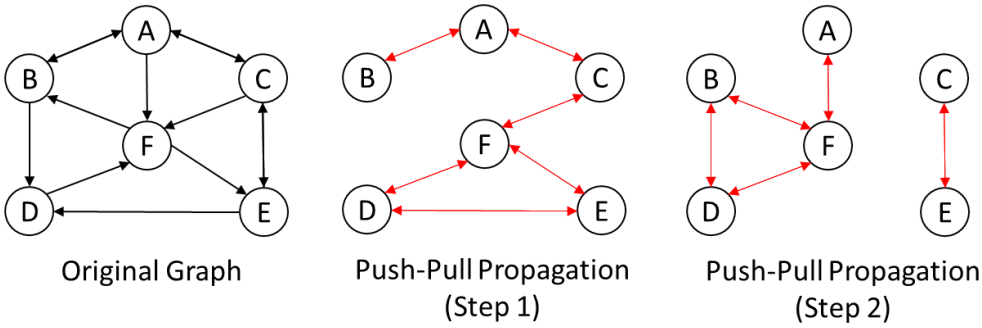


Figure 53. Push-Pull model diffusion links example. In both steps, links have been selected randomly from the adjacent nodes. More configurations are possible.

We adapt that last version for our experiments. Figure 53 shows an example of the selection of neighbours, and the flow of information in this algorithm. Every piece of received information that has not been shared is automatically discarded by the user. The diffusion ends when there are not new pieces of information to propagate.

6.4.2 Data

Once we have defined the design of our experiments, we have to select the data for them. Differently from the experiments in chapter 5, in the diffusion experiments we will focus on the 1 Month interactions graph. The execution of the different simulations over the rest of the retrieved work remains as future work.

The information which will be spread over the network consists of a selection of the tweets published by each user. For selecting those tweets, first, we remove all the retweets in the dataset: only the original tweets created for each user remain. Then, a temporal split of the tweet set is made, and we only retain those tweets in the last four days of the dataset. Finally, as we want to check the diversity of the network in terms of hashtags, we remove all the tweets which do not contain any hashtag. We finally obtain a set of 85,116 tweets which will be spread through the network.

6.4.3 Re-rankers

As we stated before, a simple way to check the effect of measure the effect of certain structural diversity metrics in the diffusion of the information consists in enhancing the value of that metric, for example, using greedy rerankers as the ones explained in section 6.2. This is the approach we take in our experiments, so we have selected three structural diversity metrics to study: clustering coefficient, modularity and community in-Gini. The three re-rankers have been applied over the outcome of the matrix factorization algorithm for implicit feedback (see section 3.4.2), since it is the algorithm which obtains better results for precision in the selected graph:

- **Clustering Coefficient:** This metric is one of the most well-known and studied metrics in Social Network Analysis, and it gives a value of the transitivity of the network. The structural diversity of the network increases inversely to the value of this metric, so, instead of enhancing this metric, we enhance its inverse.

  This metric cannot be expressed in terms of individual values, so we have used the global greedy enhancement algorithm explained in section 6.2.1. In Figure 54, we show how the precision and the value of clustering coefficient change for different values of the λ parameter of the re-ranker.

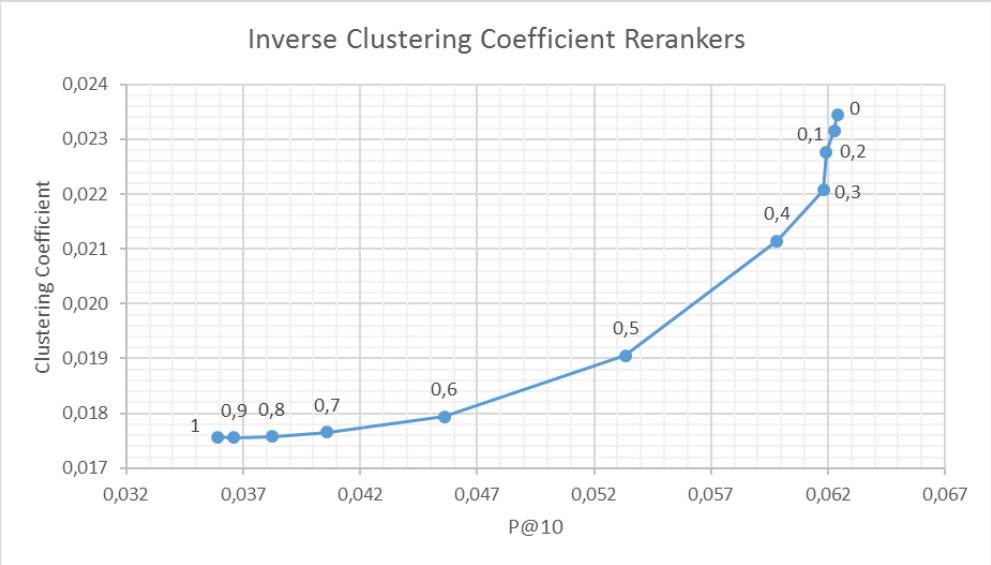


Figure 54. Clustering Coefficient and P@10 values for Inverse Clustering Coefficient Rerankers

- **Modularity:** We want to know the effects of recommending weak ties to the user over the diffusion of information in the network. This metric acts similarly to clustering coefficient in terms of structural diversity: the greater the value of the metric, the smaller the diversity of the network.

  Although this metric cannot be expressed in terms of individual values, in previous chapter, we saw that the number of weak links and the value of modularity are inversely correlated. The number of weak ties can be computed as an aggregation of individual metrics:

$$WT = |\{(u,v) \in E | comm(u) \neq comm(v)\}| = \sum_{u \in \mathcal{U}} \sum_{v \in \Gamma(u)} (1 - \delta(comm(u), comm(v))$$

  so we can use the greedy MMR approach for enhancing the number of weak ties and thus diminishing the modularity. From the three community detection approaches in the previous section, we have focused on the Louvain approach. Figure 55 shows the evolution of the values of the precision and the modularity of this reranker using different values for the trade-off parameter.

- **Community in-Gini:** This metric represents another approach to the study of the recommendation of weak links to the target users, but, instead of just maximizing the number of weak ties, distributing links equally between the different

communities. Similarly to the inverse modularity re-rankers, we use the Louvain communities for computing this metric.

This metric has been directly enhance using the global greedy approach, and Figure 56 shows the evolution of this metric and the precision for several values of λ.

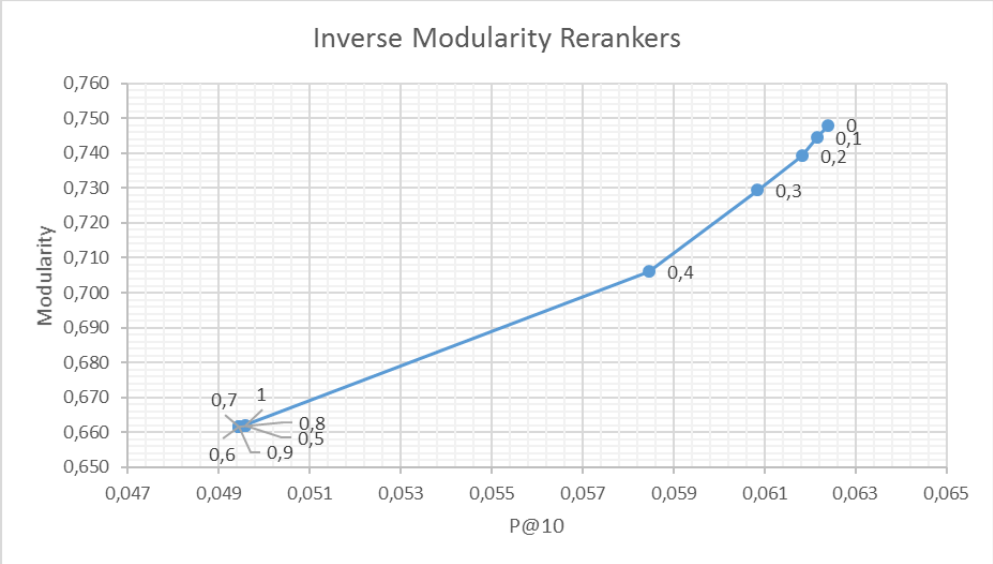


Figure 55. Modularity and P@10 values for Inverse Modularity Rerankers

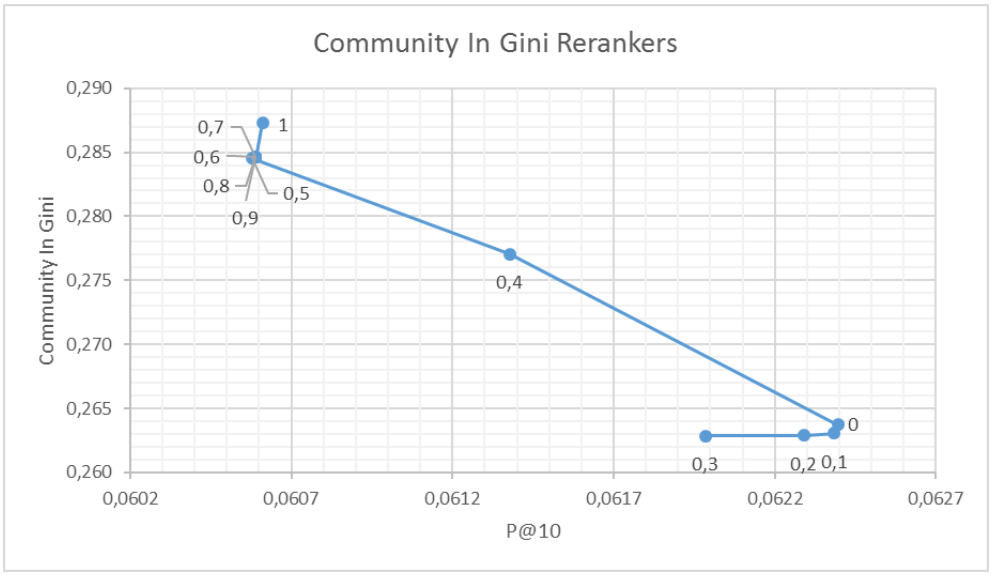


Figure 56. Community In-Gini and P@10 values for the Community In-Gini rerankers

6.5 Results

In this section, we analyze the different results we have obtained in our study of the information diffusion in the Twitter network. We divide the results in two parts: first, we will discuss the results obtained for the speed, and then, we will focus on the information diversity.

The results of the simulations have a high variance between executions, so, for paliating the possible effects of that variance, we have executed each simulations 20 times, and the results are the average values for the metrics over the different

simulations. In the cases where we show metrics related to re-rankers, we show two different variants for each one: a re-ranker which almost focuses on the enhance of the global structural diversity metric ($\lambda = 0.9$), and another one which gives similar importance to the precision and the diversity ($\lambda = 0.4$).

6.5.1 Speed

First, we study the differences between the two studied diffusion models in terms of the speed. Results for the comparison of five different recommendation algorithms (Adamic/Adar, BIR, ImplicitMF, Popularity and Random) are shown in Figure 57 and Figure 58 respectively for the Twitter and Push-Pull protocols.

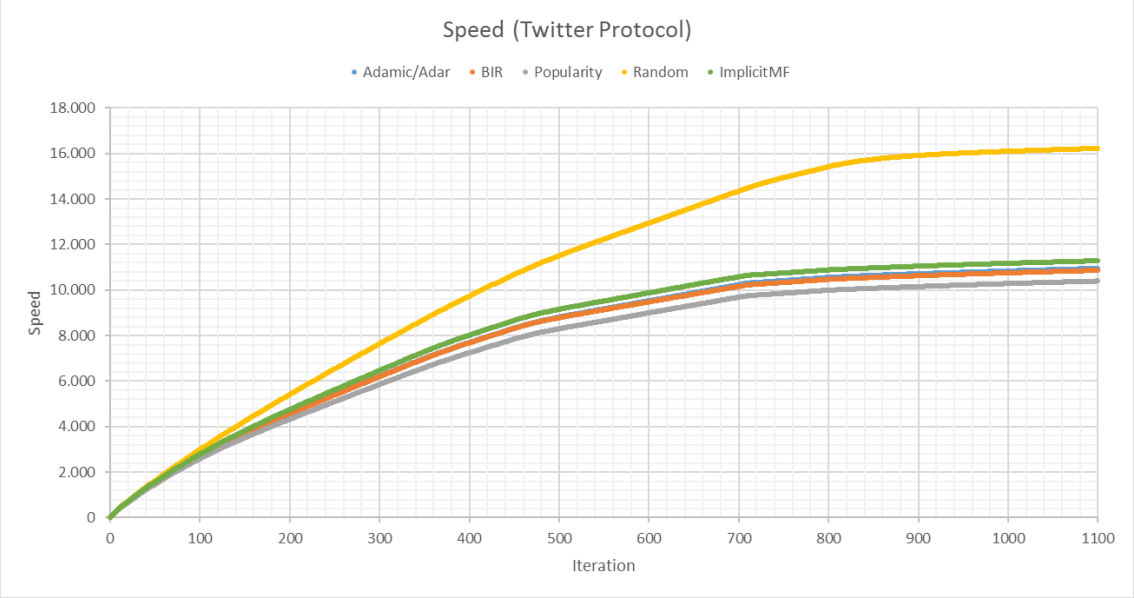


Figure 57 Diffusion speed (Twitter protocol).

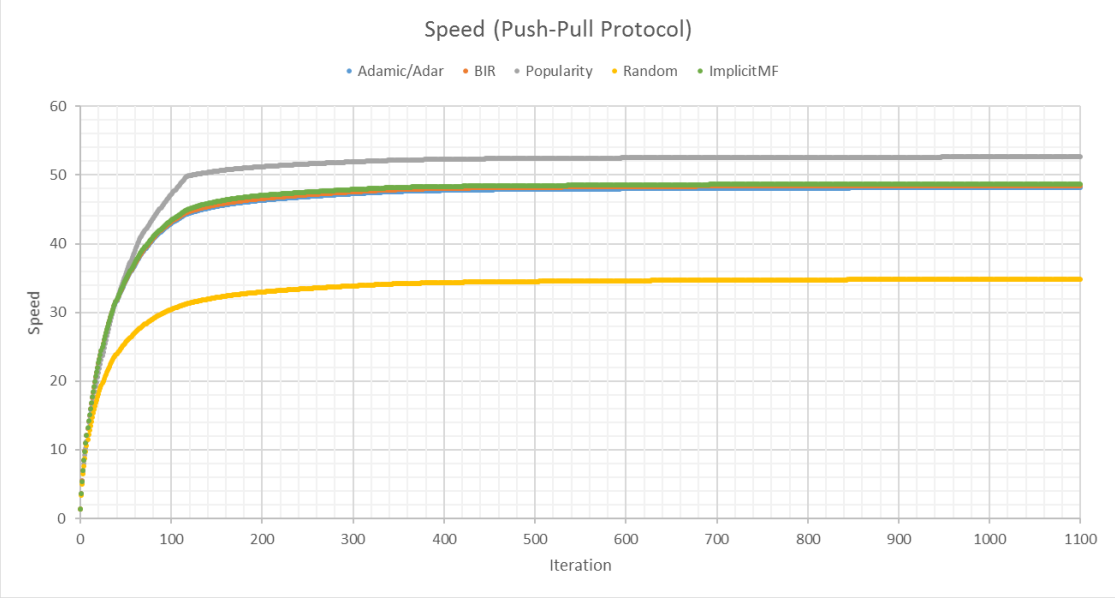


Figure 58. Diffusion speed (Push-Pull protocol)

An immediate observation is difference in the values for both protocols: Twitter protocols obtains much higher values for the metric than the Push-Pull protocol. That difference can be explained by the nature of the models: in the Twitter protocol, each

user receives information coming from all his followees (which are, at least, the 10 users from the recommendation). However, in the Push-Pull model, each user receives information from a maximum of two different users (the selected user, and if exists, a user that has selected to propagate information to the user). Since the information comes from a smaller set of users, the speed for the Push-Pull model is severely limited, which explains the differences between the speeds of both protocols.

Observing the different graphics, it is important to notice that Random and Popularity algorithms obtain the highest and lowest speed in both protocols. Random is the best in the Twitter protocol, but the worst in the Push-Pull model, while the diffusion over the Popularity extended graph is the slowest in the Twitter model, but the fastest in the Push-Pull one.

In the case of the Push-Pull model, our results are similar to the ones demonstrated by Doerr et al. (2012): in that article, they demonstrated that a single piece of information spread faster in real graphs and in preferential attachment networks (Barabási et al. 1999), characterized for a highly skewed degree distribution, that in random graphs, where the degree followed a binomial distribution. In fact, they theorized that nodes with small degree are key for the speed of the diffusion using that protocol.

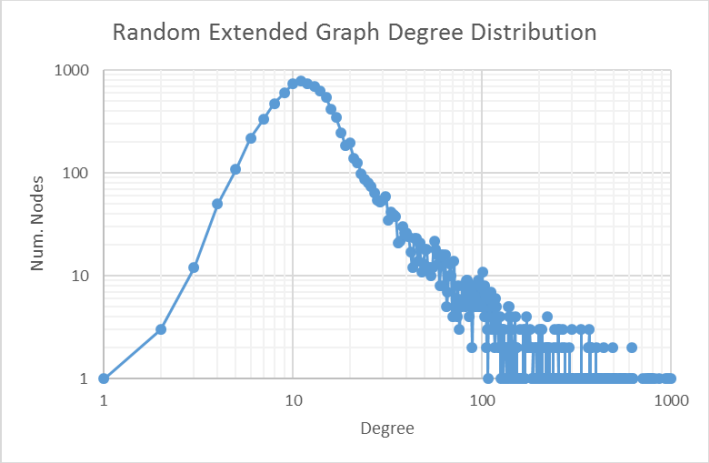


Figure 59. Degree distribution for the expanded graph for the Random algorithm

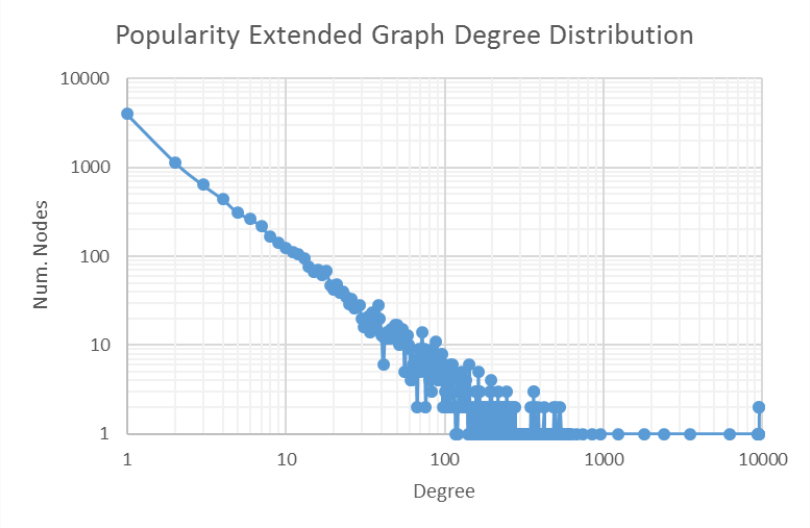


Figure 60. Degree distribution for the extended graph for Popularity algorithm

Analyzing the extended graph in-degree distributions, we observe that the Random algorithm softens the in-degree distribution of the original graph, reducing the number of nodes with small in-degree, as it can be seen in Figure 59, while the Popularity one mantains the in-degree of those algorithms, increasing only the in-degree of the most popular nodes, as it is shown in Figure 60. This seems to agree with the previous theory, since the speed of the diffusion is maximized when there are more nodes with small in-degree.

The speed of the information flow over the Popularity extended graph using the Twitter model seems to be slowed by the tweets which are not among the top 10 most popular ones: in that graph, every user is following that set of nodes. Since the degree distribution is highly skewed, most part of the information the less popular nodes receive come from the popular users. That causes that those users are more likely to propagate information which comes from the popular user to their followers. Since the rest of the users already have that information, it is possible that they had already propagated it, so a fraction of the repropagated tweets is discarded, slowing the diffusion. This does not happen in the Random extended graph: since every user follows 10 randomly selected neighbors, information that arrives to each user is different from the information which arrives to others, preventing the overlap of information, and thus, fastening the flow of information.

Next, we want to know how does the structural diversity of the network affect the speed of the diffusion. For that,we simulate the information diffusion using the expanded graphs for the different rerankers.

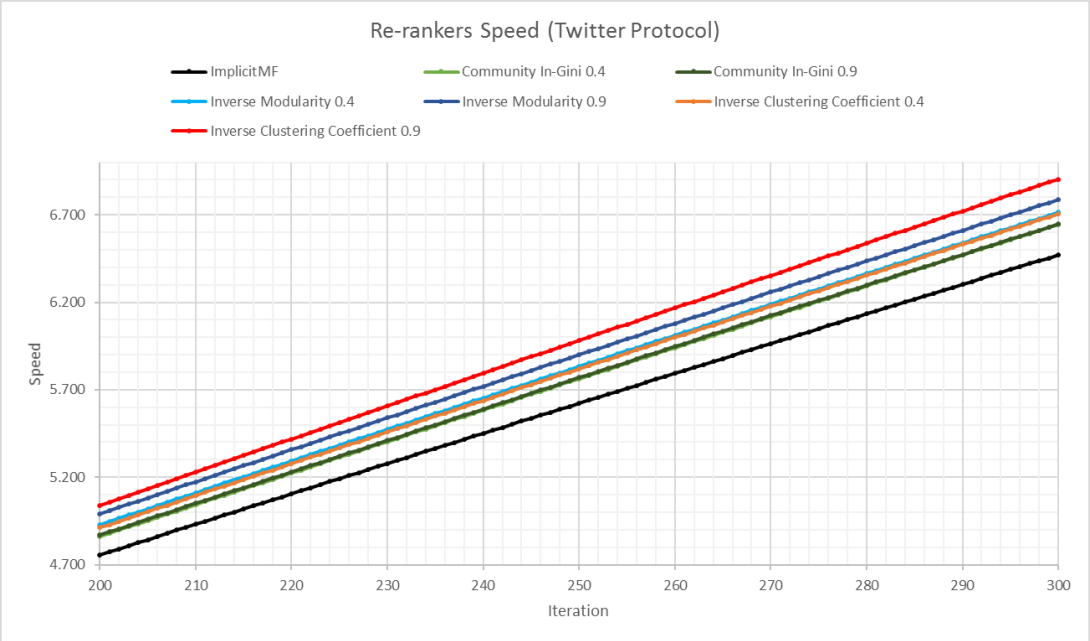


Figure 61. Diffusion speed for the different ImplicitMF re-rankers (Twitter protocol)

In Figure 61 and Figure 62 we show the speed for the different re-rankers in simulations for both communication models. In order to appreciate the differences between the algorithms, we only show the interval between 200 and 300 simulation steps, although the same behavior has been observed for the rest of the simulation. We notice that, with the decrease of modularity or clustering coefficient, the speed of the diffusion seems to increase in both Twitter and Push-Pull models: all the four re-rankers shown in the graph overcome the speed of the baseline. The same does not happen with

the community in-Gini metric: enhancing the metric does not show any effect on the Twitter protocol, and the effects of the metric over the speed are not clear: although both community in-Gini re-rankers obtain lower values for the metric than the baseline, the relation between the metric and the speed does not seem to be inversed, since the re-ranker which most enhances the speed of the algorithm ($\lambda = 0.9$) is faster than the other one ($\lambda = 0.4$).

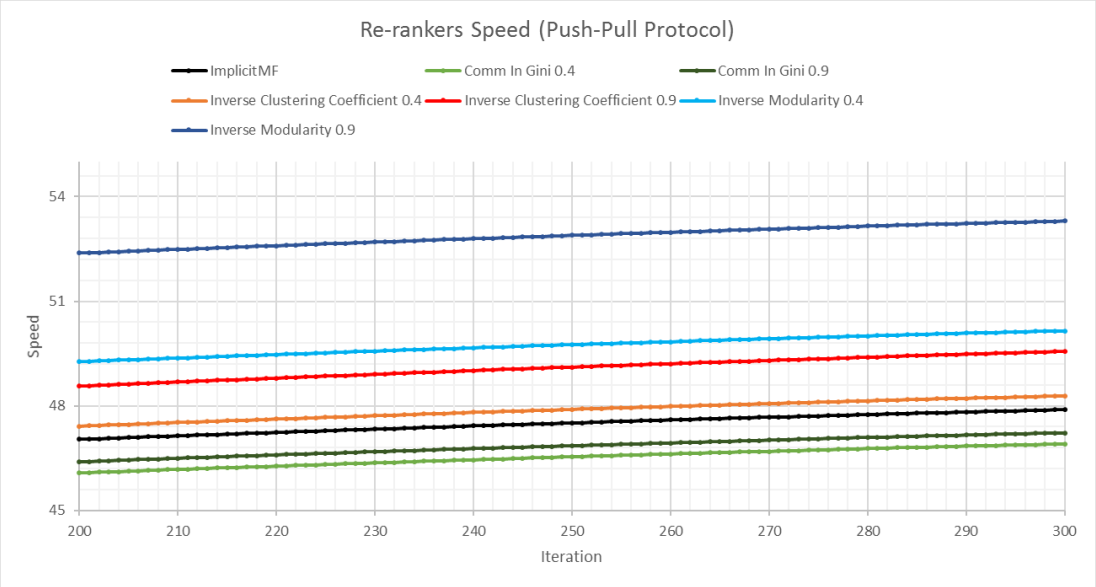


Figure 62. Diffusion speed for the different ImplicitMF re-rankers (Push-Pull protocol)

In conclusion, decreasing the values of modularity and clustering coefficient leads to an enhancement of the diffusion speed in the network.

6.5.2 Information diversity

Once we have studied the speed of the diffusion, we analyze the diversity of the information in the network. We want to know if a more structurally diverse recommendation leads to a more diverse flow of information through the network. In order to check that, we will study the diversity metrics over the simulations.

Figure 63 shows the values for the Hashtag-Global Gini for the first 400 steps of the simulations using the Twitter model. Using that protocol, all the re-rankers obtain better results than the ImplicitMF baseline. In that figure, we show that all the re-rankers overcome the baseline in terms of information diversity, specially both Community In-Gini rerankers and Inverse Clustering Coefficient rerankers. Minimizing the clustering coefficient of the network seems to achieve the better results, but with a serious problem: the accuracy of the recommendations nearly halves. Community in-Gini obtains slightly worse results, but it does not lose much precision. Similar results are obtained for the Hashtag-User Gini metric, as it is observed in Figure 64.

Information diversity, as well as the speed, depends on the communication protocol. Figure 65 and Figure 66 show the values of the Hashtag-Global Gini and Hashtag-User Gini for the simulations with the Push-Pull protocol. In those graphs, we observe that community in-Gini is the only enhanced property which clearly obtains better results than the baseline, although the differences between the two rerankers have been decreased. In the case of modularity, the relation between the number of weak ties in the network and the diversity of information seems to be greatly reduced. Other re-

rankers, like the modularity ones, decrease the diffusion of information as the metric is enhanced. Similar results are shown for the Hashtag-User Gini metric.

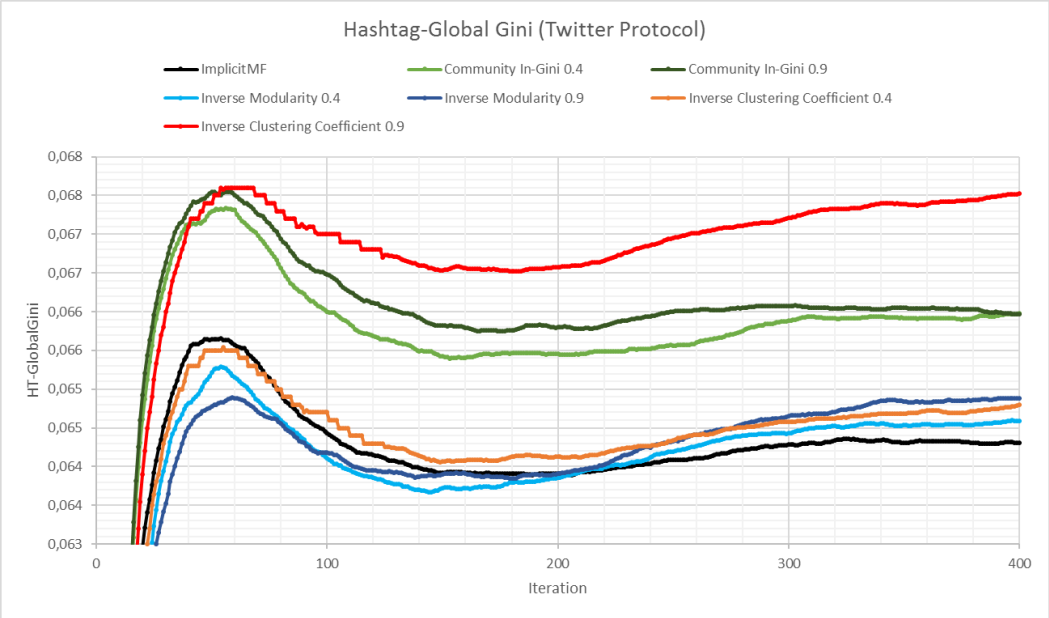


Figure 63. Hashtag-Global Gini results for the different rerankers (Twitter protocol)

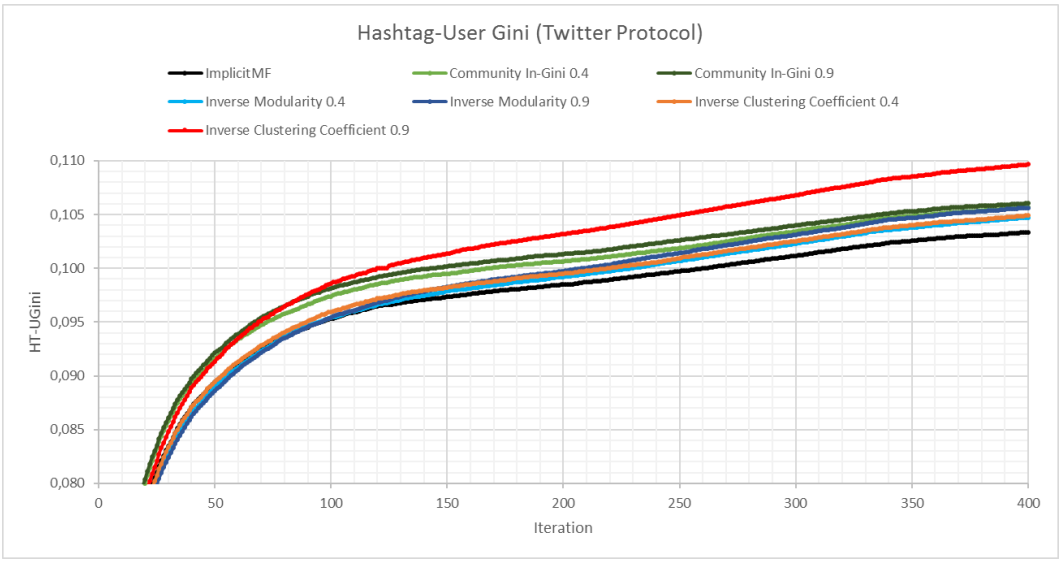


Figure 64. Hashtag-User Gini results for the different rerankers (Twitter protocol)

6.5.3 Conclusions

To sum up, we have observed that the enhancement of structural diversity measures for recommender systems usually has effects over the flow of information that spreads over the network. In general, we the different observed effects depend on two factors the enhanced metric and the diffusion protocol. However, we have found several effects which are consistent over the two studied protocols: first, reducing the clustering coefficient of the network and its modularity increase the speed of the diffusion: information reaches other users faster; second, equally distributing the links between the different communities helps paliating the filter bubble effect, since the diversity of information is enhanced.

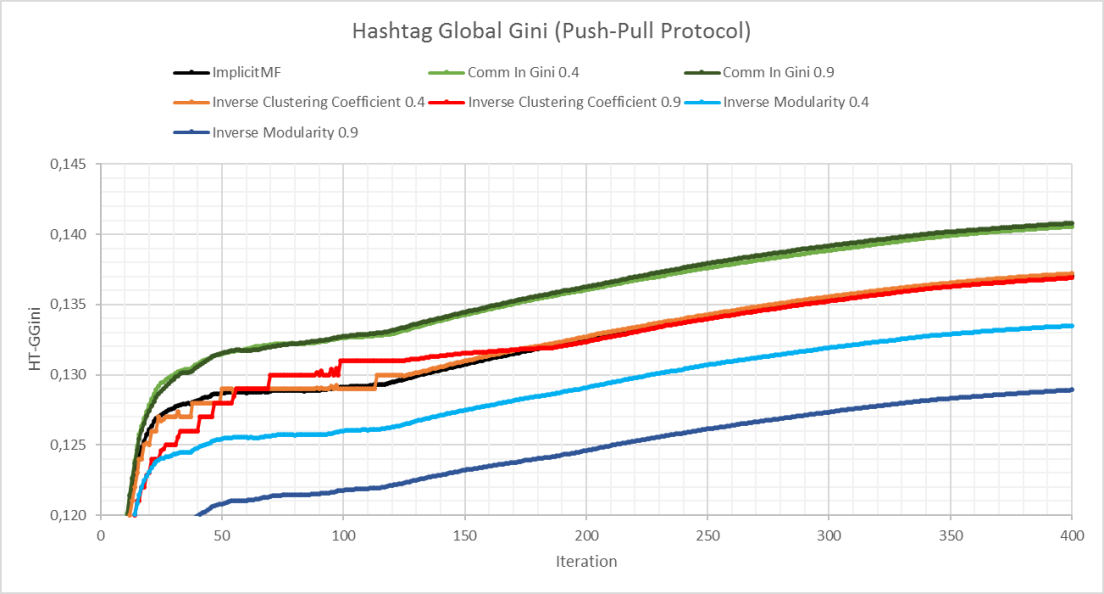


Figure 65. Hashtag-Global Gini results for the different rerankers (Push-Pull protocol)

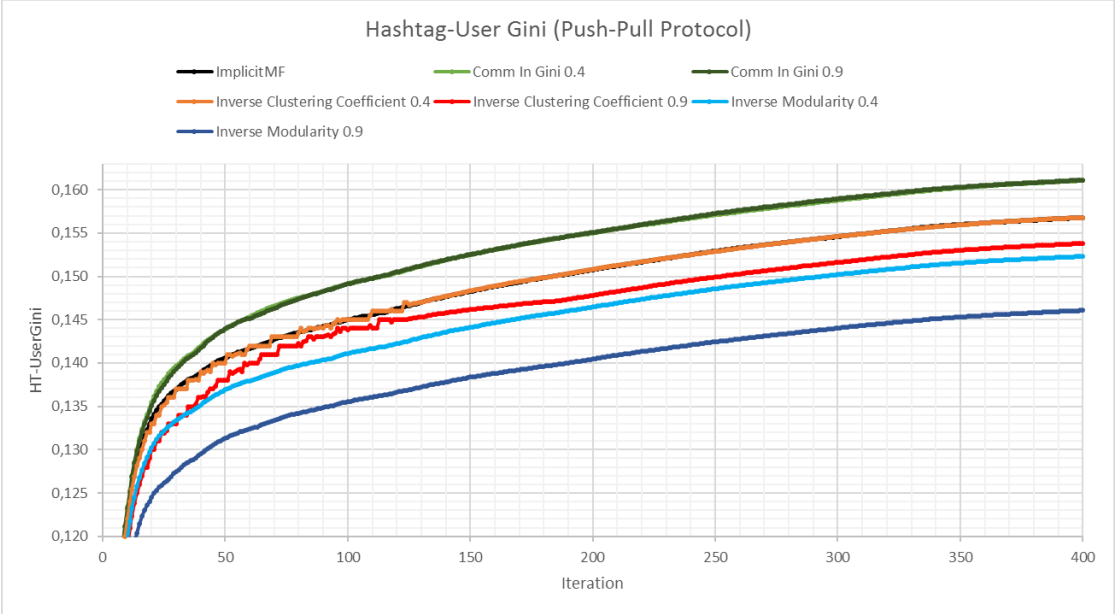


Figure 66. Hashtag-User Gini results for the different rerankers

Finally, we want to answer the remaining research question: **is it useful to recommend weak links?** From the results above, the answer to that question would be positive: at least, the recommendation of weak ties increases the speed of the diffusion, so the information reaches to more users in less time, and, if the links to other communities are similarly distributed among them, then, the information that arrives to different users is, in average, more diverse.

7. Conclusions

7.1 Summary and contributions

In the present work, we have studied the contact recommendation problem, with a focus on the effects that the different techniques may have on the shape and evolution of the social network. The evaluation and measurement of those effects considers novel perspectives, such as the novelty and diversity of the recommendations, and the effects the recommendations may have on the different properties of the network, as measured by such metrics as the clustering coefficient or the modularity of the network. We have also studied the correspondence between such structural effects on the network and the properties of the information which is propagated across users, mainly considering the diversity and the speed of the flow of information through the network.

We have analyzed the effects of several contact recommendation methods. For selecting those algorithms, we have thoroughly studied the literature related to the problem of recommending users. We have studied several fields including link prediction and classical recommendation. We have adapted and implemented the most important and effective approaches we have found. We also propose new algorithms, including adaptations of widely-known Text Information Retrieval techniques, (for example, BM25 or Query Likelihood), optimized versions of Web-oriented algorithms, such as Pure Personalized PageRank, or new algorithms like Average Cosine Similarity or Centroid Cosine Similarity inspired in prior work in the field.

For the data needs of our analysis, we have gathered two different samples from the Twitter social network. Each one of the samples contains an explicit graph of the network (known as the follows graph), and another graph reflecting the interactions (retweet, reply, mention) between the different users in the network. The sampling of the graph has been oriented to retrieving the interaction graphs (i.e. the sampling procedure consists on a traversal through such graphs), using a variant of the so-called snowball sampling technique. We differentiate both graphs by how the retrieved information has been selected: one of them has retrieved all the interactions between the crawled users during a temporal interval of time (a month), and the other one only retrieved the interactions which were made in the last 200 published tweets.

In our experiments, we first compared the different algorithms in terms of the accuracy of the recommendations, using ranking-oriented metrics such as precision, recall and nDCG. We have found that the different algorithms behave similarly in terms of accuracy in three of the four studied graphs. Although there are many differences between the studied graphs, we have found that the different versions we have proposed for the BM25 algorithm achieve high accuracy values in all of them. Also, classical recommendation techniques, such as matrix factorization methods and neighborhood-based collaborative filtering methods, are also very good. These algorithms usually work better, in our experiments, than the personalized SALSA algorithm, reported to be used by Twitter in its Who-To-Follow system.

We have also compared the different possible configurations in terms of the directions of the edges for common neighbors approaches like FOAF, Adamic-Adar and

BM25. We have found that candidate users are usually best characterized by their incoming neighborhood. We have not clearly observed which neighborhood represents better the target user: in some cases the in-neighborhood works best, in others the union of both incoming and outgoing neighbors works better.

Beyond the accuracy criterion, we have studied other properties of recommendation techniques in terms of novelty and diversity perspectives. For that, we adapted to our problem metrics from the recommendation field such as the popularity complement and the Gini coefficient. We have furthermore studied aspect-based diversity metrics, adapted from Information Retrieval, taking the communities they belong to as the equivalent of aspects. We also studied the structural diversity of the algorithms, using metrics which come from the social network analysis field, as well as other novel metrics, like community Gini.

The study of all those perspectives has given as a result that best results are generally achieved by recommending random users, and in some cases, like modularity or embeddedness, by algorithms similar to popularity-based recommendation. Also, most accurate approaches like BM25 or ImplicitMF obtain values for the different metrics which are far from optimal. Aspect-based diversity metrics make an exception to this, since they are highly correlated to the accuracy metrics.

Among the most accurate approaches, we have found that, in terms of novelty and diversity metrics, among the most accurate algorithms, we have found that the different BM25 variants recommend users more equally, ImplicitMF differs most from popularity, and personalized SALSA provides, in general, more novel and diverse recommendations for the users. In terms of structural diversity, personalized SALSA outstands in terms of modularity and clustering coefficient. ImplicitMF obtains very high values (and therefore bad values) for modularity, but it provides a good balance between precision and community Gini metrics (i.e. it is the algorithm which best distributes links between communities). ImplicitMF, as well as user-based kNN are also good in terms of distance metrics.

For all the different evaluation perspectives, we have observed differences which depend on the studied graph. However, we have not been able to find clear differences between the explicit graphs and the interaction graphs in terms of the recommendation results.

Finally, we have studied the effects that recommendations have in the flow of information of the network, focusing on the diffusion speed and the diversity of the information. For that aim, we have simulated the flow of information over the training graph of the network, with the addition of the top new links proposed by the recommendation techniques to each user. Since different communication protocols may be differently affected by the recommendation, we have used two different dynamics: the first one, represents the usual Twitter information exchange, through the user timelines, and the other one, the push-pull protocol, corresponds to communication via private messages.

Rather than directly testing the effects of each individual algorithm, we have analyzed the effects of enhancing several structural diversity properties on the ranking of the ImplicitMF recommendations. Since most of the structural diversity properties are globally defined for the network, and cannot be defined in terms of individual values for each user, we have developed a novel greedy approach for enhancing global properties of the recommendation, which swaps the positions of two elements in the

recommendation ranking if the swap improves a target function that combines the diversity metric and the recommendation score.

We have focused our study in three different structural diversity properties: clustering coefficient, modularity and community Gini. We found that reducing the clustering coefficient and the modularity of the network increases the speed of the information flow for both studied protocols. We also discovered that equally distributing the links between communities (enhancing the community Gini) provides more diverse information for the different users in the network. Finally, we have analyzed the utility of the recommendation of weak links, defined as links between users in different communities. Since both modularity and community Gini were related to them, we have found that recommending weak links is useful, since they increase the speed and the diversity of the information diffusion.

7.2 Future work

There are many posibilities for delving into the contact recommendation problem and the effects of the different approaches on the evolution of social networks. In this section, we mention some of them:

First, the recommendation of users in social networks is still an open field, and many new algorithms and techniques are in development. Although we have studied and analyzed the most important approaches, as well as some new ones, a possible way of expanding our reseach consists in trying and researching new algorithms, in particular supervised methods, which have had a somewhat lesser development yet in this area. In addition to the new algorithms, several new structural diversity metrics may be added to the study, like the closeness of the users or the betweenness of the recommended edges, which may help us to better underestan the behavior of the different contact recommendation algorithms.

Our research so far has only focused on the Twitter social network, so another posibility lies on the study of other social networks like Tumblr, Facebook or Linked which show different properties from Twitter. Over those networks, we might check if the results are consistent with the ones reported in the present work.

Also, it would be interesting to study the effects that the sample graph produce on the effectiveness of the recommendations. As we have seen in the four different graphs we have studied, the properties of a single algorithm might vary depending on the graph we recommend users on. Understanding the characteristics of the networks which affect these properties may help us to provide better recommendations on different networks, as well as improving the desired global properties of the network.

Focusing on the study of the information diffusion of the networks, our work can be easily expanded with the study of the effects caused by other structural diversity metrics like recommendation distance or edge betweenness. In addition to these metrics, we could also measure the effects of enhancing traditional novelty and diversity metrics like Gini coefficient or popularity complement on the information diffusion. In addition, more communication protocols like the Independent Cascade Model might be tested in the simulations.

Also, on our experiments on information diffusion we have this far only used one of the four retrieved graphs. We plan to check the consistency of our results in the remaining graphs in future work.

Bibliography

Adamic, L.A., Adar, E. Friends and Neighbors on the Web. Social Networks, 25(3), July 2003, pp. 211-230.

Adomavicius, G., Tuzhilin, A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), June 2005, pp. 734-749.

Adomavicius, G., Kwon, Y. Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques. IEEE Transactions on Knowledge and Data Engineering 24(5), May 2012, pp. 896-911.

Agrawal, R., Gollapudi, S., Halverson, A., Ieong, S. Diversifying Search Results. 2nd Annual ACM International Conference on Web Search and Data Mining (WSDM 2009), Barcelona, Spain, February 2009, pp. 5-14.

Al Hasan, M., Chaoji, V., Salem, S., Zaki, M. Link Prediction Using Supervised Learning. Workshop on Link Analysis, Counterterrorism and Security in 2006 SIAM Conference on Data Mining (SDM 2006), Bethesda, Maryland, USA, April 2006.

Amatriain, X. Mining Large Streams of User Data for Personalized Recommendations. ACM SIGKDD Explorations Newsletter, 14(2), December 2012, pp. 37-48.

Aral, S. The Future of Weak Ties. American Journal of Sociology, 121 (6), May 2016, pp. 1931-1939.

Baeza-Yates, R., Ribeiro-Neto, B. Modern Information Retrieval: The Concepts and Technology Behind Search, 2nd Edition. Addison Wesley, 2010.

Backstrom, L., Leskovec, J. Supervised Random Walks: Predicting and Recommending Links in Social Networks. 4th Annual ACM International Conference on Web Search and Data Mining (WSDM 2011), Hong Kong, China, Februrary 2011, pp. 635-644.

Bakshy, E., Messing, S., Adamic, L.A. Exposure to ideollogically diverse news and opinion on Facebook. Science 348 (6239), June 2015, pp. 1130-1132.

Barabási, A., Albert, R. Emergence of Scaling in Random Networks, Science 286(5439), November 1999, pp. 509-512.

Bellogín, A., Wang, J., Castells, P. Bridging Memory-Based Collaborative Filtering and Text Retrieval. Information Retrieval 16(6), December 2013, pp. 697-724.

Blondel, V.D., Guillaume, J-L., Lamiotte, R., Lefebvre. Fast Unfolding of Communities in Large Networks. Journal of Statistical Mechanics: Theory and Experiment, 2008, October 2008.

Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., Wagner, D. On Modularity Clustering. IEEE Transactions on Knowledge and Data Engineering, 20(2), February 2008, pp. 172-188.

Brin, S., Page, L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. 7th Annual International Conference on World Wide Web (WWW 1998), Brisbane, Australia, April 1998, pp. 107-117.

Carbonell, J., Goldstein, J. The User of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. 21st Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR 1998), Melbourne, Australia, August 1998, pp. 335-336.

Castells, P., Hurley, N.J., Vargas, S. Novelty and Diversity in Recommender Systems. In Ricci et al. (2015), pp. 881-918.

Chapelle, O., Metlzer, D., Zhang, Y., Grinspan, P. Expected Reciprocal Rank for Graded Relevance. 18th Annual ACM International Conference on Information and Knowledge Management (CIKM 2009), Hong Kong, China, November 2009, pp. 621-630.

Chapelle, O., Shihao, J., Liao, C., Velipasaoglu, E., Lai, L., Wu, S. Intent-based Diversification of Web Search Results: Metrics and Algorithms. Information Retrieval 14(6), December 2011, pp. 572-592.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P. SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16(1), January 2002, pp. 321-357.

Chebotarev, P.Y., Shamis, E.V. The Matrix-Forest Theorem and Measuring Relations in Small Social Groups. Automation and Remote Control, 58(9), 1997, pp.1505-1514.

Clarke, C.L.A., Kolla, M., Cormack, G.V., Vechtomova, O., Ashkan, A., Büttcher, S., MacKinnon, I. Novelty and Diversity in Information Retrieval Evaluation. 31st Annual International ACM Conference on Research and development in Information Retrieval (SIGIR 2008), Singapore, Singapore, July 2008, pp. 659-666.

Clauset, A., Newman, M.E.J., Moore, C. Finding Community Structure in Very Large Networks. Physical Review E 70(6), December 2004.

Cremonesi, P., Koren, Y., Turrin, R. Performance of Recommender Algorithms on Top-N Recommendation Tasks. 4th Annual International ACM Conference on Recommender Systems (RecSys 2010), Barcelona, Spain, September 2010, pp. 39-46.

Daly, E.M., Geyer, W., Millen, D.R. The Network Effects of Recommending Social Connections. 4th Annual International ACM Conference on Recommender Systems (RecSys 2010), Barcelona, Spain, September 2010, p. 301-304.

Danklemann, P., Goddard, W., Swart, C.S. The Average Eccentricity of a Graph and its Subgraphs. Utilitas Mathematica 65, May 2004, pp. 41-51.

Das, G., Koudas, N., Papagelis, M., Puttaswamy, S. Efficient Sampling of Information in Social Networks. 2008 ACM Workshop in Search in Social Media (SSM 2008), Napa Valley, California, USA, October 2008, pp. 67-74.

Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., Terry, D. Epidemic Algorithms for Replicated Database Maintenance. 6th Annual ACM Symposium on Principles of Distributed Computing (PODC 1987). Vancouver, British Columbia, Canada, August 1987, pp. 1-12.

De Meo, P., Ferrara, E., Fiumara, G., Provetti, A. On Facebook, Most Ties are Weak Communications of the ACM 57(11), November 2014, pp.78-84.

Doerr, B., Fouz, M., Friedrich, T., Social networks spread rumors in sublogarithmic time, 43rd Annual ACM Symposium on Theory of Computing (STOC 2011), June 2011, pp. 21-30.

Durkheim, E. De la division du travail social: étude sur l'organisaion des sociétés supérieures. Alcan, 1893.

Easley, D., Kleinberg, J. Networks, Crowds and Markets. Reasoning about a Highly Connected World. Cambridge University Press, Cambridge University Press, 2010.

Erdös, P. Rényi, A. On Random Graphs. I. Publicationes Mathematicae 6, 1959, pp. 290-297.

Gao, S., Denoyer, L., Gallinari, P. Temporal Link Prediction by Integrating Content and Structure Inforamtion. 20th ACM International Conference on Information and Knowledge Management (CIKM 2011), Glasgow, Scotland, UK, October 2011, pp. 1169-1174.

Goel, A., The "Who-To-Follow" System at Twitter: Algorithms, Impact and Further Research. 23rd Annual International Conference on World Wide Web (WWW 2014), Seoul, South Korea, April 2014, industry track.

Goel, A., Gupta, P., Sirois, J., Wang, D., Sharma, A., Gurumurthy, S. The Who-To-Follow System at Twitter: Strategy, Algorithms, and Revenue Impact. Interfaces 45(1), February 2015, pp. 98-107.

Goldbeck, J. Generating predictive movie recommendations from trust in social networks. 4th International Conference in Trust Management (iTrust 2006), Pisa, Italy, May 2006, pp. 93-104.

Goldberg, D., Nichols, D., Oki, B.M., Terry, D. Using Collaborative Filtering to Weave an Information Tapestry, Communications of the ACM 35(12), December 2012, pp. 61-70.

Goldenberg, J., Libai, B., Muller, E. Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth. Marketing Letters 12(3), 2001, pp. 211-223.

Golder, S.A., Yardi, S., Marwick, A., boyd, d. A Structural Approach to Contact Recommendation in Online Social Networks. Workshop on Search in Social Media in 32nd ACM International Conference on Research and Development in Information Retrieval (SIGIR 2009), Boston, Massachussets, USA, July 2009.

Goodman, L.A. Snowball sampling. Annals of Mathematical Statistics, 31(1), 1961, pp. 148-170.

Granovetter, M., The Strength of Weak Ties. American Journal of Sociology 78(6), May, 1973, pp. 1360-1380.

Guille, A. Hacid, H., Favre, C. Zighed, D.A. Information Diffusion in Online Social Networks: A Survey. ACM SIGMOD Record 42(2), May 2013, pp. 17-28.

Gupta, P., Goel, A., Lin, J., Sharma, A., Wang. D., Zadeh, R. WTF: The Who to Follow Service at Twitter. 22nd Annual International Conference on World Wide Web (WWW 2013), Rio de Janeiro, Brazil, May 2013, pp. 505-514.

Guy, I. Social Recommender Systems. In Ricci et al. (2015), pp. 512-543.

Hannon, J., Bennet, M., Smyth, B. Recommending Twitter Users to Follow Using Content and Collaborative Filtering Approaches. 4th Annual International ACM Conference on Recommender Systems (RecSys 2010), pp. 199-206.

Hethcote, H.W. The Mathematics of Infectious Diseases. SIAM Review 42(4), December 2000, pp. 599-653.

Hu, Y., Koren, Y., Volinsky, C. Collaborative Filtering for Implicit Feedback Datasets. 8th Annual IEEE International Conference on Data Mining (ICDM 2008), Pisa, Italy, December 2008, pp. 263-272.

Huang, X., Tiwari, M., Shah, S. Structural Diversity in Social Recommender Systems. 5th ACM RecSys Workshop on Recommender Systems and the Social Web (RSWeb 2013), Hong Kong, China, October 2013, pp.

Huffman, D.A. A Method for the Construction of Minimum-Redundancy Codes. Proceedings of the Institute of Radio Engineers, 40(9), September 1952, pp. 1098-1101.

Järvelin, K., Kekäläinen, J. IR evaluation methods for retrieving highly relevant documents. 23rd Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR 2000), Athens, Greece, July 2000, pp. 41-48.

Jelinek, F., Mercer, R.L. Interpolated estimation of Markov source parameters from sparse data. In E. S. Gelsema and L. N. Kanal, editors, Pattern Recognition in Practice . North-Holland, Amsterdam, 1980, pp. 381-402.

Katz, L. A new status index derived from sociometric analysis. Psychometrika 18(1), March 1953, pp. 39-43.

Kempe, D., Kleinberg, J., Tardos, E. Maximizing the Spread of Influence Through a Social Network. 9th Annual International ACM Conference on Knowledge Discovery and Data Mining (SIGKDD 2003), Washington, DC, USA, August 2003.

Kim, Y., Shim, K. TWITOBI: A Recommendation System for Twitter Using Probabilistic Modelling. IEEE 11th International Conference on Data Mining (ICDM 2011), Vancouver, Canada, December 2011, pp. 340-349.

Kleinberg, J.M., Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A.S. (1999A). The Web as a Graph: Measurements, Models and Methods. 5th Annual International Conference on Computing and Combinatorics (COCOON 1999), Tokyo, Japan, July 1999.

Kleinberg, J.M. (1999B) Authoritative Sources in a Hyperlinked Environment. Journal of the ACM 46(5), September 1999, pp. 604-632.

Konstas, I. Stathopoulos, V., Jose, J.M., On Social Networks and Collaborative Recommendation. 32nd International Conference on Research and Development in Information Retrieval (SIGIR 2009). Boston, Massachussets, USA, July 2009, pp. 195-202.

Koren, Y., Bell, R., Volinsky, C. Matrix Factorization Techniques for Recommender Systems. IEEE Computer 42(8), August 2009, pp. 30-37.

Krishnamurthy, B., Gill, P., Arlitt, M. A Few Chirps About Twitter. 1st Workshop on Online Social Networks, Seattle, WA, USA, August 2008, pp. 19-24.

Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tomkins, A. Upfal, E. Stochastic Models for the Web Graph. 41st Annual Symposium on Foundations of Computer Science, Redondo Beach, CA, USA, November 2000, pp. 57-65.

Kwak, H., Lee, C., Park, H., Moon, S. What is Twitter, a Social Network or a News Media?. 19th Annual International Conference on World Wide Web, Raleigh, North Carolina, USA, April 2010, pp. 591-600.

Lee, J.H. Analyses of Multiple Evidence Combination. 20th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR 1997), Philadelphia, Pennsylvania, USA, July 1997, pp. 267-276.

Leicht, E.A., Holme, P., Newman, M.E.J. Vertex similarity in networks. Physical Review E 73(2), 026120, February 2006.

Lempel, R., Moran, S. SALSA: The Stochastic Approach for Link-Structure Analysis. ACM Transactions on Information Systems 19(2), April 2001, pp. 131-160.

Leskovec, J., Faloutsos, C. Sampling from Large Graphs. 12th Annual International ACM Conference on Knowledge Discovery and Data Mining (SIGKDD 2006), Philadelphia, Philadelphia, USA, August 2006.

Leskovec, J., Kleinberg, J., Faloutsos, C. Graph Evolution: Densification and Shrinking Diameters. ACM Transactions on Knowledge Discovery from Data (TKDD), 1 (1), March 2007, article num. 2.

Leskovec, J., Backstrom, L., Kumar, R., Tomkins, A. Microscopic Evolution of Social Networks. 14tj ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2008). Las Vegas, Nevada, USA, August 2008.

Liben-Nowell, D., Kleinberg, J. The Link Prediction Problem for Social Networks. Journal of the American Society for Information Science and Technology 58(7), May 2007.

Lichtenwalter, R.N., Lussier, J.T., Chawla, N.V. New Perspectives and Methods in Link Prediction. 16th Annual ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2010), Washington, DC, USA, July 2010, pp. 243-252.

Lü, L., Jin, C., Zhou, T. Similarity Index Based on Local Paths for Link Prediction of Complex Networks. Physical Review E 80(4) : 046122, October 2009, pp

Lü, L., Zhou. T. Link Prediction in Complex Networks: A survey. Physica A: Statistical Mechanics and its Applications, 390(6), March 2011, pp. 1150-1170.

Ma, H., Yang, H., Lyu, M.R., King, I. SoRec: Social Recommendation Using Probabilistic Matrix Factorization. 17th Annual ACM Conference on Information and Knowledge Management (CIKM 2008), Napa Valley, California, USA, October 2008, pp. 931-940.

McPherson, M, Smith-Lovin, L., Cook, J.M. Birds of a Feather: Homophily in Social Networks. Annual Review of Sociology, 27, 2001, pp. 415-444.

Menon, A.K., Elkan, C. Link Prediction Via Matrix Factorization. 2011 European Conference on Machine Learning And Knowledge Discovery in Databases (ECML/PKDD 2011), Athens, Greece, September 2011, Vol. Part II, pp. 437-452.

Meyer, C.D. The Role of the Group Generalized Inverse in the Theory of Finite Markov Chains. SIAM Review 17(3), July 1975, pp. 443-464.

Milgram, S. The Small World Problem. Psychology Today 1(1), May 1967, pp. 61-67.

Myers, S.A., Sharma, A., Gupta, P., Lin, J. Information Network or Social Network?: The Structure of the Twitter Follow Graph. 23rd Annual International Conference on World Wide Web (WWW 2014), Seoul, Korea, April 2014, pp. 493-498.

Newman, M.E.J. (2001A). Clustering and Preferential Attachment in Growing Networks. Physical Review Letters E, 64(025102), April 2001.

Newman, M.E.J. (2001B). Scientific Collaboration Networks. II. Shortest paths, weighted networks and centrality. Physical Review E 64 : 016132, June 2001.

Newman, M.E.J. The Structure and Function of Complex Networks. SIAM Review 45(2), 2003, pp. 167-256.

Newman, M.E.J., Girvan, M. Finding and Evaluating Community Structure in networks. Physical Review E 69(2): 026113, February 2004.

Newman, M.E.J. Finding Community Structure in Networks Using the Eigenvectors of Matrices. Physical Review E 74(3), September 2006.

Newman, M.E.J. Networks. An introduction. Oxford University Press, 2010.

Nguyen, T.T., Hui, P., Harper, F.M., Terveen, L., Konstan, J.A. Exploring the Filter Bubble: The Effect of Using Recommender Systems on Content Diversity. 23rd International Conference on World Wide Web (WWW 2014), Seoul, Korea, April 2014, pp. 667-686.

Ning, X., Desrosiers, C., Karypis, G. A Comprehensive Survey of Neighborhood-Based Recommendation Methods. In Ricci et al. (2015), pp. 37-76.

Orman, G.K., Labatut, V., Cherifi, H. On Accuracy of Community Structure Discovery Algorithms. Journal of Convergence Information Technology, 6 (11), December 2011, pp. 283-292.

Pariser, E. The Filter Bubble: How the New Personalized Web is Changing What We Read and How We Think. Penguin Press, May 2011.

Parotsidis, N., Pitoura, E., Tsaparas, P. Centrality-Aware Link Recommendations. 9th ACM International Conference on Web Search and Data Mining (WSDM 2016). San Francisco, California, USA, February 2016, pp. 503-512.

Pizzato, L., Rej, T., Chung, T., Koprinska, I. Kay, J. RECON: A Reciprocal Recommender for Online Dating. 4th ACM Annual International Conference on Recommender Systems (RecSys 2010), Barcelona, Spain, September 2010, pp. 207-214.

Ponte, J. M. Croft, W. B. A language modeling approach to information retrieval. 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1998). Melbourne, Australia, August 1998, pp. 275-281.

Rabanny, R., Takaffoli, M., Fagnan, J., Zaiane, O.R., Campello, R.J.G.B. Relative Validity Criteria for Community Mining Algorithms. 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012), Istanbul, Turkey, August 2012, pp. 258-265.

Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D. Detecting and Identifying Communities in Networks. Proceedings of the National Academy of Sciences of the USA 101(9), January 2004, pp. 2658-2664.

Raghavan, U.N, Albert, R., Kumara, S. Near Linear Time Algorithm to Detect Community Structures in Large-Scale Networks. Physical Review E 76 (3), September 2007

Ravasz, E., Somera, A.L., Mongru, D.A., Oltvai, Z.N., Barabási, A.-L. Hierarchical Organization in Metabolic Networks, Science 297, August 2002.

Ricci, F., Rokach, L., Shapira, B. Recommender Systems Handbook 2nd. Edition. Springer, 2015.

Robertson, S. E., Sparck Jones, K. Relevance weighting of search terms. Journal of the Association for Information Science and Technology (JASIST) 27(3), May 1976, pp. 129-146.

Robertson, S.E., The Probability Ranking Principle in IR. Journal of Documentation 33, 1977, pp. 294-304.

Rogers, E.M. Diffusion of Innovations. The Free Press, 1962.

Rosvall, M., Bergstrom, C.T. Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences of the United States of America (PNAS), 105(4), January 2008, pp. 1118-1123.

Salakhutdinov, R., Mnih, A. Probabilistic Matrix Factorization. 21st Annual Conference on Neural Information Processing Systems (NIPS 2007), Vancouver, Canada, December 2007, pp. 1257-1264.

Salton, G., Wong, A., Yang, C.S. A vector space for automatic indexing. Communications of the ACM 18(11), November 1975, pp. 613-620.

Salton, G., McGill, M.J. Introduction to Modern Information Retrieval. McGraw-Hill, 1984.

Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.T. Application of Dimensionality Reduction in Recommender System – A Case Study. 2nd Workshop on Web Mining for e-Commerce (WebKDD 2000), Boston, Massachussets, USA, August 2000.

Shani, G., Gunawardana, A., Evaluating Recommender Systems. In Ricci (2015), pp. 265-308.

Sørensen, T.J., A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons. Biologiske skrifter 5(4), 1948

Sparck Jones, K., Walker, S., Roberton S.E. A Probabilistic Model of Information Retrieval: Development and Comparative Experiments. Information Processing and Management 36. February 2000, pp. 779-808 (part 1), pp. 809-840 (part 2).

Su, J., Sharma, A., Goel, S. The Effect of Recommendations on Network Structure. 25th Annual International Conference on World Wide Web (WWW 2016), Montreal, Québec, Canada, April 2016.

Tang, J., Hu, X., Liu, H. Social Recommendation: A Review. Social Network Analysis and Mining 3(4), December 2013, pp. 1113-1133.

Tönnies, F. Gemeinschaft und Gesellschaft. Fues' Verlag. 1887.

Ugander, J., Karrer, B, Backstrom, L., Marlow, C. The Anatomy of the Facebook Social Graph. Arxiv Computer Research Repository, 1111.4503, November 2011.

Vahabi, H., Koutsopoulos, I., Gullo, F., Halkidi, M. DifRec: A Social-Diffusion-Aware Recommender System. 24th ACM Internatioanl Conference on Information and Knowledge Management, Melbourne, Australia, October 2015, pp. 1481-1490.

Vargas, S., Castells, P. Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems. 5th Annual International ACM Conference on Recommender Systems (RecSys 2011), Chicago, Illinois, USA, October 2011, pp. 109-116.

Vargas, S., Castells, P. Improving Sales Diversity by Recommending Users to Items. 8th Annual International ACM Conference on Recommender Systems (RecSys 2014), Foster City, Silicon Valley, California, USA, Octobe 2014, pp. 145-152.

Watts, D.J., Strogatz, S.H. Collective Dynamics of 'Small-World' Networks. Nature 393, June 1998, pp. 440-442.

White, S., Smyth, P. Algorithms for Estimating Relative Importance in Networks. 9th Annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003), Washington D.C., USA, August 2003.

Zafarani, R., Ali Abbasi, M., Liu, H. Social Media Mining. An Introduction. Cambridge University Press, 2014.

Zhai, C.X., Cohen, W.W., Lafferty, J. Beyond Independent Relevance: Methods and Evaluation Metrics for Subtopic Retrieval. 26th Annual International ACM SIGIR Conference on Research and development in Information Retrieval (SIGIR 2003), Toronto, Canada, July 2003, pp. 10-17.

Zhou, T., Lü, L., Zhang, Y. Predicting missing links via local information. The European Physical Journal B, 71, October 2009, pp. 623-630.

Annex I: Derivations

In this annex, we show the construction of some of the formula which we have used in our experiments.

Contact Recommendation Algorithms

This section contains the derivations of some of the proposed recommendation algorithms, which have their origins on previous ones.

BM25

The original formula for BM-25 recommendation method is

$$f_u(v) = \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{(k+1)}{k\left(1 - b + \frac{b|\Gamma(v)|}{\text{avg}_{v' \in \mathcal{U}}(|\Gamma(v')|)}\right) + 1} RSJ'(w)$$

where

$$RSJ'(w) = \log_2 \frac{|U| - |\Gamma(w)| + 0.5}{|\Gamma(w)| + 0.5}$$

Extreme-BM-25

Extreme BM 25 is defined as the limit version of BM 25 when the free parameter k tends to infinity. The derivation for this formula is shown below:

$$\begin{aligned}
\lim_{k\to\infty} f_u(v) &= \lim_{k\to\infty} \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{(k+1)}{k\left(1 - b + \frac{b|\Gamma(v)|}{\text{avg}_{v' \in \mathcal{U}}(|\Gamma(v')|)}\right) + 1} RSJ'(w) \\
&= \sum_{w \in \Gamma(u) \cap \Gamma(v)} \lim_{k\to\infty} \frac{(k+1)}{k\left(1 - b + \frac{b|\Gamma(v)|}{\text{avg}_{v' \in \mathcal{U}}(|\Gamma(v')|)}\right) + 1} RSJ'(w) \\
&= \sum_{w \in \Gamma(u) \cap \Gamma(v)} \lim_{k\to\infty} \frac{1 + \frac{1}{k}}{\left(1 - b + \frac{b|\Gamma(v)|}{\text{avg}_{v' \in \mathcal{U}}(|\Gamma(v')|)}\right) + \frac{1}{k}} RSJ'(w) \\
&= \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{RSJ'(w)}{1 - b + \frac{b|\Gamma(v)|}{\text{avg}_{v' \in \mathcal{U}}(|\Gamma(v')|)}}
\end{aligned}$$

Binary Independent Ranking (BIR)

It is usual to see Binary Independent Ranking (Robertson & Sparck Jones 1976) as a previous version of BM 25 model. However, if we make b tend to 0 in this second model, we retrieve the original BIR probabilistic model, as we show next.

$$\lim_{b\to 0^+} f_u(v) = \lim_{b\to 0^+} \sum_{w\in\Gamma(u)\cap\Gamma(v)} \frac{(k+1)}{k\left(1-b+\frac{b|\Gamma(v)|}{\text{avg}_{v'\in\mathcal{U}}(|\Gamma(v')|)}\right)+1} RSJ'(w)$$
$$= \sum_{w\in\Gamma(u)\cap\Gamma(v)} \lim_{b\to 0^+} \frac{(k+1)}{k\left(1-b+\frac{b|\Gamma(v)|}{\text{avg}_{v'\in\mathcal{U}}(|\Gamma(v')|)}\right)+1} RSJ'(w)$$
$$= \sum_{w\in\Gamma(u)\cap\Gamma(v)} \frac{k+1}{k+1} RSJ'(w) = \sum_{w\in\Gamma(u)\cap\Gamma(v)} RSJ'(w)$$

The same happens if we make b tend to 0 in ExtremeBM25.

If we make b tend to 1, we obtain the following:

$$\lim_{b\to 1^-} f_u(v) = \lim_{b\to 1^-} \sum_{w\in\Gamma(u)\cap\Gamma(v)} \frac{(k+1)}{k\left(1-b+\frac{b|\Gamma(v)|}{\text{avg}_{v'\in\mathcal{U}}(|\Gamma(v')|)}\right)+1} RSJ'(w)$$
$$= \sum_{w\in\Gamma(u)\cap\Gamma(v)} \lim_{b\to 1^-} \frac{(k+1)}{k\left(1-b+\frac{b|\Gamma(v)|}{\text{avg}_{v'\in\mathcal{U}}(|\Gamma(v')|)}\right)+1} RSJ'(w)$$
$$= \sum_{w\in\Gamma(u)\cap\Gamma(v)} \frac{k+1}{k\frac{|\Gamma(v)|}{\text{avg}_{v'\in\mathcal{U}}(|\Gamma(v')|)}+1} RSJ'(w)$$
$$= \frac{k+1}{k\frac{|\Gamma(v)|}{\text{avg}_{v'\in\mathcal{U}}(|\Gamma(v')|)}+1} \sum_{w\in\Gamma(u)\cap\Gamma(v)} RSJ'(w)$$

If we make $k \to \infty$, then:

$$f_u(v) = \frac{\text{avg}_{v'\in\mathcal{U}}(|\Gamma(v')|)}{|\Gamma(v)|} \sum_{w\in\Gamma(u)\cap\Gamma(v)} RSJ'(w) \propto \frac{BIR(v)}{Pref.Att.(v)}$$

PurePersonalized PageRank

In this section, we define the Pure Personalized PageRank method as a probability. We have to take into account that the probability for a random walker of being in a certain node in a certain time depends only on the previous position of the walker in the graph:

$$p(v|t) = \sum_i p(v|w, t-1)p(w|t-1)$$

We want to compute the stationary probability for that random walker, so we suppose that the probability is time-independent, and we decompose it on the probability of transition in the probability of traversing an edge (as PageRank was originally developed for web graphs, we maintain the notation, and consider traversing an edge as clicking a hyperlink) and the probability of teleporting to another node.

$$p_u(v) = \sum_w p_u(v|w)p_u(w)$$
$$= \sum_w [p_u(v|w, edge)p_u(edge|w)$$
$$+ p_u(v|w, teleport)p_u(teleport|w)]\, p_u(w)$$

We establish r as the teleporting rate for nodes which are not sinks. If the random walker reaches a sink, he always teleports:

$$p_u(edge|w) = \begin{cases} 1-r & \text{if } |\Gamma_{out}(w) - \{u\}| > 0 \\ 0 & \text{if } |\Gamma_{out}(w) - \{u\}| = 0 \end{cases}$$

$$p(teleport|w) = \begin{cases} r & \text{if } |\Gamma_{out}(w) - \{u\}| > 0 \\ 1 & \text{if } |\Gamma_{out}(w) - \{u\}| = 0 \end{cases}$$

We only allow reaching the origin node by teleport, even when there is a link to it:

$$p_u(v|w, edge) = \begin{cases} \dfrac{1}{|\Gamma_{out}(w) - \{u\}|} & \text{if } w \in \Gamma_{in}(v) \\ 0 & \text{if not} \end{cases}$$

$$p_u(u|w, click) = 0$$

We only allow teleporting to the origin node when the actual node is not a sink. In other case, it could randomly teleport to another user in the network.

$$p(v|w, edge) = \begin{cases} 0 & \text{if } v \neq u \text{ and } |\Gamma_{out}(w) - \{u\}| > 0 \\ 1 & \text{if } v = u \text{ and } |\Gamma_{out}(w) - \{u\}| > 0 \\ \dfrac{(1-r)}{|\mathcal{U}| - 1} & \text{if } v \neq u \text{ and } |\Gamma_{out}(w) - \{u\}| = 0 \\ r & \text{if } v = u \text{ and } |\Gamma_{out}(w) - \{u\}| = 0 \end{cases}$$

Now, we divide the derivation in two separate cases: The probability of the target user, and the probability of the rest of them:

$$\begin{aligned} p_u(u) &= \sum_w p_u(u|w) p_u(w) \\ &= \sum_w [p_u(u|w, edge) p_u(edge|w) \\ &+ p_u(u|w, teleport) p_u(teleport|w)] \, p_u(w) \\ &= \sum_w p_u(u|w, edge) p_u(edge|w) p_u(w) \\ &+ \sum_w p(u|w, teleport) p_u(teleport|w) p_u(w) \\ &= \sum_w p_u(u|w, teleport) p_u(teleport|w) p_u(w) \\ &= \sum_{w:|\Gamma_{out}(w) - \{u\}| > 0} p_u(u|w, teleport) p_u(teleport|w) p_u(w) \\ &+ \sum_{w:|\Gamma_{out}(w) - \{u\}| = 0} p_u(u|w, teleport) p_u(teleport|w) p_u(w) \\ &= \sum_w 1 \cdot r \cdot p_u(w) + \sum_{w:|\Gamma_{out}(w) - \{u\}| = 0} r \cdot 1 \cdot p_u(w) = r \sum_w p_u(w) = r \end{aligned}$$

In the other case:

$$p_u(v) = \sum_w p_u(v|w)p_u(w)$$
$$= \sum_w p_u(v|w, edge)p_u(edge|w)p_u(w)$$
$$+ \sum_w p_u(v|w, teleport)p_u(teleport|w)p_u(w)$$
$$= \sum_{w:|\Gamma_{out}(w)-\{u\}|>0} p_u(v|w, edge)p_u(edge|w)p_u(w)$$
$$+ \sum_{w:|\Gamma_{out}(w)-\{u\}|=0} p_u(v|w, teleport)p_u(teleport|w)p_u(w)$$
$$= (1-r) \sum_{w\in\Gamma_{in}(v)} \frac{p_u(w)}{|\Gamma_{out}(w)-\{u\}|} + \frac{(1-r)}{N-1} \sum_{w:|\Gamma_{out}(w)-\{u\}|=0} p_u(w)$$

$$p_u(v) = \sum_w p_u(v|w)p_u(d_i)$$
$$= \sum_i \big[p\big(d_j|d_i, edge\big)P(edge|d_i)$$
$$+ p\big(d_j|teleport\big)p(teleport|d_i)\big]\, p(d_i) =$$
$$= (1-r) \sum_{d_i\in\Gamma_{in}(d_j)} \big(1-\delta_{uj}\big)\frac{p_u(d_i)}{|\Gamma_{out}(d_i)-\{u\}|}$$
$$+ \sum_{d_i:|\Gamma_{out}(d_i)-\{u\}|=0} \delta_{uj}p_u(d_i) + r \sum_{d_i:\Gamma_{out}(d_i)>0} \delta_{uj}\, p_u(d_i) =$$

Then:

$$p_u(v) = \begin{cases} r & \text{if } v = u \\ (1-r) \sum_{w\in\Gamma_{in}(v)} \frac{p_u(w)}{|\Gamma_{out}(w)-\{u\}|} + \frac{(1-r)}{N-1} \sum_{w:|\Gamma_{out}(w)-\{u\}|=0} p_u(w) & \text{if } v \neq u \end{cases}$$

Metrics

In this annex, we will include how some of the evaluation metrics have been defined.

Modularity

Modularity has been traditionally described in the literature for undirected graphs (Newman 2010). Since we use directed ones in our experiments, in this section, we describe how we have obtained the equations for that case. The procedure is similar to the one explained in Newman (2010), ch. 7 for the undirected modularity.

First of all, we define the formula for unnormalized modularity, as follows:

$$\mathrm{Q}(G, C) = h - \mathbb{E}[h]$$

where h is the number of edges between elements that share the same nominal value, and $\mathbb{E}[h]$ stands for the number of expected edges between elements of the same class in a multigraph where the edges are placed at random, considering the degree distributions of the studied graph. In the directed graph, the total number of edges that run between vertices of the same class is

$$\sum_{i \to j} \delta(c_i, c_j) = \sum_{ij} A_{ij} \delta(c_i, c_j)$$

Now, we have to calculate the expected number of edges between vertices if edges are placed at random. Given a node with in-degree $|\Gamma_{in}(j)|$, the probability that a particular edge is attached to that vertex is $|\Gamma_{in}(j)|/m$. Given that there are $|\Gamma_{out}(i)|$ edges that start in node i, the expected number of edges between vertices i and j is $|\Gamma_{in}(j)||\Gamma_{out}(i)|/m$. With that in mind, it is clear that the expected number of edges between all pairs of vertices of the same type is:

$$\sum_{ij} \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m} \delta(c_i, c_j)$$

With those two values, it is possible to obtain the value for the unnormalized modularity:

$$\begin{aligned} Q(G, C) = h - \mathbb{E}[h] &= \sum_{ij} A_{ij} \delta(c_i, c_j) - \sum_{ij} \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m} \delta(c_i, c_j) \\ &= \sum_{ij} \left(A_{ij} - \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m} \right) \delta(c_i, c_j) \end{aligned}$$

As a convention, this value is divided by the number of edges, so the unnormalized modularity is:

$$Q(G, C) = \frac{1}{m} \sum_{ij} \left(A_{ij} - \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m} \right) \delta(c_i, c_j)$$

As we want the metric to be comparable between different divisions of the network, we want this value to be equal to 1 when the division of the network is perfect (there are no edges between two different classes). This ideal value is reached when every link in inside a community (i.e. $\delta(c_i, c_j) = 1$ if $A_{ij} = 1$). Using this in the equation above, then:

$$\begin{aligned} Q_{\max}(G, C) &= \frac{1}{m} \sum_{ij} \left(A_{ij} - \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m} \right) \delta(c_i, c_j) \\ &= \frac{1}{m} \sum_{ij} A_{ij}^2 - \frac{1}{m} \sum_{ij} \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m} \delta(c_i, c_j) \\ &= \frac{1}{m} \sum_{ij} A_{ij} - \frac{1}{m} \sum_{ij} \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m} \delta(c_i, c_j) \\ &= \frac{1}{m} \left(m - \sum_{ij} \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m} \delta(c_i, c_j) \right) \end{aligned}$$

So the final modularity value is:

$$\mathrm{mod}(G) = \frac{\sum_{ij} \left(A_{ij} - \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m} \right) \delta(c_i, c_j)}{m - \sum_{i,j} \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m} \delta(c_i, c_j)} = \frac{\sum_{ij} A_{ij} \delta(c_i, c_j) - K(G, C)}{m - K(G, C)}$$

where

$$K(G,C) = \sum_{ij} \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m} \delta(c_i, c_j)$$

Assortativity

The modularity of scalar values measures how correlated is some scalar value at the ends of the links of a graph. It is computed as the Pearson correlation of the values at each extreme. Let x_i be the scalar value for vertex i. Consider the pairs of values (x_i, x_j) First of all we want to compute their covariance over all edges in the graph. For that, it is necessary to define the mean μ of the values at each end of an edge:

$$\mu_{out} = \frac{\sum_{ij} A_{ij} x_i}{\sum_{ij} A_{ij}} = \frac{\sum_i |\Gamma_{out}(i)| x_i}{\sum_i |\Gamma_{out}(i)|} = \frac{1}{m} \sum_j |\Gamma_{in}(j)| x_j$$

$$\mu_{in} = \frac{\sum_{ij} A_{ij} x_j}{\sum_{ij} A_{ij}} = \frac{\sum_j |\Gamma_{in}(j)| x_j}{\sum_i |\Gamma_{in}(j)|} = \frac{1}{m} \sum_j |\Gamma_{in}(j)|\, x_j$$

Once we have obtained them, we can compute the mentioned covariance.

$$\begin{aligned} Q(G) = cov(X_{in}, X_{out}) &= \mathbb{E}[(X_{in} - \mathbb{E}[X_{in}])(X_{out} - \mathbb{E}[X_{out}])] \\ &= \mathbb{E}[X_{in} X_{out}] - \mathbb{E}[X_{in}]\mathbb{E}[X_{out}] \\ &= \frac{1}{m} \sum_{ij} A_{ij} x_i x_j - \frac{1}{m^2} \left(\sum_i |\Gamma_{out}(i)| x_i \right) \left(\sum_j |\Gamma_{in}(j)| x_j \right) \end{aligned}$$

$$= \frac{1}{m} \sum_{i \to j} x_i x_j - \frac{1}{m^2} \left(\sum_i |\Gamma_{out}(i)| x_i \right) \left(\sum_j |\Gamma_{in}(j)| x_j \right)$$

It is possible to simplify the notation as follows:

$$Q(G) = \frac{1}{m} \sum_{ij} \left(A_{ij} - \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m} \right) x_i x_j$$

After that, we have to define the standard deviation for the scores in each extreme of the graph:

$$\sigma_{in} = \sqrt{\mathbb{E}[X_{in}^2] - \mu_{in}^2}$$

$$\mu_{in}^2 = \frac{1}{m^2} \left(\sum_j |\Gamma_{in}(j)|\, x_j \right)^2 = \frac{1}{m^2} \sum_{ij} |\Gamma_{in}(i)||\Gamma_{in}(j)| x_i x_j$$

$$\mathbb{E}[X_{in}^2] = \frac{1}{m} \sum_{ij} A_{ij}\, x_j^2 = \cdots = \frac{1}{m} \sum_j |\Gamma_{in}(j)|\, x_j^2$$

$$\sigma_{in}^2 = \frac{1}{m} \sum_{ij} \left(|\Gamma_{in}(j)| \delta_{ij} - \frac{|\Gamma_{in}(i)||\Gamma_{in}(j)|}{m} \right) x_i x_j$$

The derivation of the formula for σ_{out} is equivalent, so we are not showing it here. Then the modularity final result is the following:

$$d - assort(G) =$$

$$= \frac{\sum_{ij}\left(A_{ij} - \frac{|\Gamma_{out}(i)||\Gamma_{in}(j)|}{m}\right)x_i x_j}{\sqrt{\sum_{ij}\left(|\Gamma_{in}(j)|\delta_{ij} - \frac{|\Gamma_{in}(i)||\Gamma_{in}(j)|}{m}\right)x_i x_j}\sqrt{\sum_{ij}\left(|\Gamma_{out}(j)|\delta_{ij} - \frac{|\Gamma_{out}(i)||\Gamma_{out}(j)|}{m}\right)x_i x_j}}$$

To simplify the calculations, the easiest way to write this formula is the following one:

$$d - assort(G) =$$

$$= \frac{\sum_{i\to j} x_i x_j - \frac{1}{m}\left(\sum_i |\Gamma_{out}(i)| x_i\right)\left(\sum_j |\Gamma_{in}(j)| x_j\right)}{\sqrt{\sum_j |\Gamma_{in}(j)|\, x_j^2 - \frac{1}{m}\left(\sum_j |\Gamma_{in}(j)|\, x_j\right)^2}\sqrt{\sum_j |\Gamma_{out}(j)|\, x_j^2 - \frac{1}{m}\left(\sum_j |\Gamma_{out}(j)|\, x_j\right)^2}}$$

Now, let's apply this formula to degree modularity. There are three cases. We can compare the in-degrees, the out-degrees or the undirected degree. In this document, we only use the comparison between in degrees:

$$d - assort(G, in, in) =$$

$$= \frac{\sum_{i\to j}|\Gamma_{in}(i)||\Gamma_{in}(j)| - \frac{1}{m}\left(\sum_i |\Gamma_{in}(i)|^2\right)\left(\sum_j |\Gamma_{in}(j)||\Gamma_{out}(j)|\right)}{\sqrt{\sum_i |\Gamma_{out}(i)||\Gamma_{in}(i)|^2 - \frac{1}{m}\left(\sum_i |\Gamma_{out}(i)||\Gamma_{in}(i)|\right)^2}\sqrt{\sum_i |\Gamma_{in}(j)|^3 - \frac{1}{m}\left(\sum_i |\Gamma_{in}(j)|^2\right)^2}}$$

Annex II: Complete results

In this annex, we show the complete results for the offline experiments in chapter 5. In all the different tables in this annex, each column represents a single metric, and the gradient of colors represents how good is the value in relation to the rest of the algorithms. Green means that the algorithms obtains a good value for that metric, and red means the contrary. Cells with white letters represent the highest values of the metric in the graph. All algorithms are ordered in terms of the value of the P@10 metric.

1 Month

First, we show the different results for the 1 Month dataset.

Interactions

In this section, we show the complete results for the interactions graph of this dataset. We divide the results in 9 different tables.

First, in Table 24, we show the values of the accuracy metrics. Then, tables 25 and 26 show the values for the novelty and diversity metrics. Tables 27 and 28 display the values for embeddedness, clustering coefficient and distance metrics. Tables 29 and 30 show the modularity of the extended graphs, the number of weak ties and the degree assortativity. Finally, tables 31 and 32 show the community Gini coefficients.

Algorithm	P@10	R@10	nDCG@10	Algorithm	P@10	R@10	nDCG@10
ImplicitMF (k=280.α=150;λ=40)	**0.06123**	**0.10460**	**0.10252**	Love (Hubs;k=50;α=0.3)	0.02347	0.04319	0.04305
UserBased kNN (k=120)	0.05982	0.09787	0.10164	Popularity	0.02337	0.04093	0.04273
Personalized SALSA (Auth.;r=0.99)	0.05774	0.09899	0.09826	Preferential Attachment ($\Gamma_{\text{in/out}}(u)$)	0.02337	0.04093	0.04273
Average Cosine Similarity	0.05541	0.08435	0.08852	Hitting Time	0.02245	0.03946	0.03836
ItemBased kNN (k=300)	0.05441	0.08279	0.08710	Hub Promoted Index ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.02133	0.02783	0.03209
Centroid Cosine Similarity	0.05406	0.08241	0.08550	Hannon ($\Gamma_{in}(u)$)	0.02033	0.03235	0.03111
Local Path Index ($\Gamma_{\text{out}}(u)$;β=0.1;l=3)	0.05297	0.08320	0.08776	PageRank (r=0.1)	0.01989	0.04353	0.03844
Adamic ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.05061	0.06947	0.07434	Personalized PageRank (r=0.4)	0.01807	0.04345	0.03694
BIR ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.05049	0.07047	0.07458	Jaccard ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.01691	0.02090	0.02384
Money (k=1000;α=0.3)	0.04850	0.07677	0.07517	Sorensen ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.01691	0.02090	0.02384
FOAF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.04845	0.06597	0.07078	Hub Depressed Index ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.01687	0.02150	0.02377
BM25 ($\Gamma_{und}(u).\Gamma_{in}(v)$;$b$=0.1;$k$=1.0)	0.04657	0.06280	0.06578	TF-IDF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.01385	0.01802	0.01957
ExtremeBM25 ($\Gamma_{und}(u).\Gamma_{in}(v)$;b=0.1)	0.04515	0.06098	0.06340	Salton ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.01274	0.01649	0.01813
Resource Allocation ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.04508	0.06221	0.06604	Katz ($\Gamma_{out}(u)$;β=0.1)	0.01018	0.01712	0.01651
Pure Personalized PageRank (r=0.4)	0.04470	0.08028	0.07461	Distance (Dir.)	0.00626	0.01238	0.01002
QLJM ($\Gamma_{und}(u).\Gamma_{in}(v).\lambda$=0.1)	0.04350	0.05631	0.06200	LHN Index 1 ($\Gamma_{in}(u).\Gamma_{out}(v)$)	0.00321	0.00601	0.00514
PropFlow ($\Gamma_{\text{out}}(u)$;l=5)	0.04345	0.07681	0.07242	PMF Sigmoid (k=50.0;λ=0.01;ϕ=0.01)	0.00245	0.00440	0.00381
Personalized HITS (Auth.; r=0.99)	0.04133	0.05770	0.06400	LHN Index 2 (β=0.4)	0.00116	0.00254	0.00197
MatrixForest (α=0.001)	0.03879	0.05287	0.05483	Random	0.00061	0.00079	0.00081
Maximum Cosine Similarity	0.03027	0.05576	0.05087	PMF Basic (k=40.0;λ=0.01;ϕ=0.01)	0.00052	0.00045	0.00062
Commute Time	0.02389	0.04144	0.04044				

Table 24. Comparison of the different algorithms in terms of P@10. R@10 and nDCG@10 (1 Month interactions graph)

	Novelty		Diversity		ERR-IA			Subtopic-Recall		
Algorithm	PC	PD	ILD	Gini	Infomap	Louvain	Leading Vector	Infomap	Louvain	Leading Vector
ImplicitMF (k=280.α=150;λ=40)	0.9571	0.2005	0.2012	0.0288	0.0802	0.0721	0.0740	**0.1249**	**0.0471**	**0.1249**
UserBased kNN (k=120)	0.9459	0.2580	0.2365	0.0182	**0.0818**	**0.0800**	**0.0818**	0.1183	0.0444	0.1183
Personalized SALSA (Auth.;r=0.99)	0.9275	0.1998	0.1872	0.0115	0.0766	0.0748	0.0766	0.1184	0.0445	0.1184
Average Cosine Similarity	0.9479	0.1711	0.1027	0.0331	0.0749	0.0729	0.0749	0.1069	0.0400	0.1069
ItemBased kNN (k=300)	0.9486	0.1767	0.1136	0.0317	0.0740	0.0783	0.0802	0.1053	0.0395	0.1053
Centroid Cosine Similarity	0.9414	0.1753	0.0993	0.0206	0.0712	0.0693	0.0712	0.1046	0.0392	0.1046
Local Path Index ($\Gamma_{\text{out}}(u)$;β=0.1;l=3)	0.9209	0.1972	0.0701	0.0069	0.0718	0.0698	0.0718	0.1047	0.0394	0.1047
Adamic ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.9585	0.2282	0.2029	0.0564	0.0641	0.0614	0.0641	0.1022	0.0388	0.1022
BIR ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.9580	0.2311	0.2125	0.0502	0.0635	0.0606	0.0635	0.1031	0.0391	0.1031
Money (k=1000;α=0.3)	0.9332	0.1925	0.1635	0.0078	0.0600	0.0575	0.0600	0.1038	0.0404	0.1038
FOAF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.9594	0.2233	0.1963	0.0450	0.0617	0.0595	0.0617	0.0981	0.0372	0.0981
BM25 ($\Gamma_{und}(u).\Gamma_{in}(v)$;$b$=0.1;$k$=1.0)	0.9640	0.2033	0.1703	0.0549	0.0562	0.0535	0.0562	0.0960	0.0364	0.0960
ExtremeBM25 ($\Gamma_{und}(u).\Gamma_{in}(v)$;b=0.1)	0.9652	0.2043	0.1731	0.0542	0.0539	0.0513	0.0539	0.0945	0.0359	0.0945
RA ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.9646	0.2616	0.2528	0.0878	0.0568	0.0544	0.0568	0.0932	0.0353	0.0932
Pure Personalized PageRank (r=0.4)	0.9490	0.2606	0.3167	0.0303	0.0553	0.0523	0.0553	0.1062	0.0401	0.1062
QLJM ($\Gamma_{und}(u).\Gamma_{in}(v).\lambda$=0.1)	0.9721	0.2221	0.2008	0.1170	0.0556	0.0530	0.0556	0.0880	0.0334	0.0880
PropFlow ($\Gamma_{\text{out}}(u)$;l=5)	0.9562	0.2604	0.3007	0.0271	0.0547	0.0516	0.0547	0.1017	0.0383	0.1017
Personalized HITS (Auth.; r=0.99)	0.9048	0.3758	0.2673	0.0025	0.0549	0.0531	0.0549	0.0863	0.0324	0.0863
MatrixForest (α=0.001)	0.9787	0.2249	0.2161	0.1249	0.0490	0.0469	0.0490	0.0835	0.0317	0.0835
Maximum Cosine Similarity	0.9624	0.2069	0.1643	0.0572	0.0391	0.0383	0.0391	0.0665	0.0249	0.0665
Commute Time	0.9174	0.5972	0.5350	0.0011	0.0273	0.0253	0.0273	0.0648	0.0248	0.0648

Table 25. Comparison of the different algorithms in terms novelty and diversity (1 Month interactions graph) (1 of 2)

	Novelty		Diversity		ERR-IA			Subtopic Recall		
Algorithm	PC	PD	ILD	Gini	Infomap	Louvain	Leading Vector	Infomap	Louvain	Leading Vector
Love (Hubs;k=50;α=0.3)	0.9098	0.5789	0.1175	0.0014	0.0308	0.0301	0.0308	0.0566	0.0211	0.0566
Popularity	0.8835	0.6174	0.5247	0.0011	0.0310	0.0300	0.0310	0.0659	0.0247	0.0659
Preferential Attachment ($\Gamma_{\text{in/out}}(u)$)	0.8835	0.6174	0.5247	0.0011	0.0310	0.0300	0.0310	0.0659	0.0247	0.0659
Hitting Time	0.9181	0.6081	0.5598	0.0011	0.0256	0.0237	0.0256	0.0626	0.0239	0.0626
Hub Promoted Index ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.9857	0.2760	0.2344	0.1032	0.0308	0.0298	0.0308	0.0466	0.0174	0.0466
Hannon ($\Gamma_{in}(u)$)	0.9916	0.2074	0.1173	0.2724	0.0250	0.0240	0.0250	0.0452	0.0171	0.0452
PageRank (r=0.1)	0.9171	**0.6657**	0.7093	0.0011	0.0231	0.0206	0.0231	0.0602	0.0231	0.0602
Personalized PageRank (r=0.4)	0.9056	0.6625	**0.7155**	0.0011	0.0201	0.0191	0.0201	0.0552	0.0209	0.0552
Jaccard ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.9963	0.2299	0.2151	0.2122	0.0226	0.0217	0.0226	0.0376	0.0140	0.0376
Sorensen ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.9963	0.2299	0.2151	0.2122	0.0226	0.0217	0.0226	0.0376	0.0140	0.0376
Hub Depressed Index ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.9963	0.2352	0.2244	0.2089	0.0227	0.0218	0.0227	0.0397	0.0148	0.0397
TF-IDF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.9966	0.2596	0.2760	0.2528	0.0176	0.0171	0.0176	0.0297	0.0110	0.0297
Salton ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.9966	0.2444	0.2392	0.1658	0.0167	0.0163	0.0167	0.0272	0.0102	0.0272
Katz ($\Gamma_{out}(u)$;β=0.1)	0.9737	0.4931	0.3617	0.0142	0.0141	0.0134	0.0141	0.0270	0.0101	0.0270
Distance (Dir.)	0.9925	0.2594	0.2985	0.4805	0.0072	0.0069	0.0072	0.0183	0.0068	0.0183
LHN Index 1 ($\Gamma_{in}(u).\Gamma_{out}(v)$)	0.9984	0.2562	0.1667	0.0737	0.0040	0.0037	0.0040	0.0092	0.0035	0.0092
PMF Sigmoid (k=50.0;λ=0.01;ϕ=0.01)	0.9936	0.6004	0.6898	0.0010	0.0030	0.0027	0.0030	0.0081	0.0030	0.0081
LHN Index 2 (β=0.4)	**0.9999**	0.1912	0.0877	0.0857	0.0011	0.0012	0.0011	0.0030	0.0011	0.0030
Random	0.9980	0.6009	0.6452	**0.8486**	0.0008	0.0007	0.0008	0.0020	0.0007	0.0020
PMF Basic (k=40.0;λ=0.01;ϕ=0.01)	0.9979	0.6123	0.6290	0.0123	0.0006	0.0007	0.0006	0.0017	0.0007	0.0017

Table 26. Comparison of the different algorithms in terms novelty and diversity (1 Month interactions graph) (2 of 2)

	Clustering Coefficient		Embeddedness		Distances				
Algorithm	Global	Average	Global	Zero	ASL	Avg. Ecc.	Diameter	Rec. Dist	Edges ∞
ImplicitMF (k=280.α=150;λ=40)	0.0954	0.2843	0.0245	26.557	3.2663	6.7466	8	2.10	6,852
UserBased kNN (k=120)	0.0919	0.3353	0.0221	28.388	3.0509	6.3164	7	2.28	7,610
Personalized SALSA (Auth.;r=0.99)	0.0955	0.3835	0.0222	23.746	2.9905	6.2094	8	2.31	**25,433**
Average Cosine Similarity	0.1107	0.3039	0.0263	26.402	3.2765	6.5762	9	2.08	7,002
ItemBased kNN (k=300)	0.1033	0.2830	0.0249	27.476	3.2068	7.0921	9	2.36	7,039
Centroid Cosine Similarity	0.1124	0.3340	0.0234	25.585	3.1922	6.5231	8	2.15	7,012
Local Path Index ($\Gamma_{\text{out}}(u)$;β=0.1;l=3)	0.1046	0.4260	0.0217	22.888	3.0792	6.4424	8	2.15	7,015
Adamic ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.1173	0.4469	0.0286	20.618	3.1534	5.9391	8	2.09	7,014
BIR ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.1178	0.4321	0.0276	21.137	3.1636	6.0440	8	2.09	7,013
Money (k=1000;α=0.3)	0.1099	0.4062	0.0202	25.015	3.1044	6.4871	8	2.30	6,958
FOAF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.1205	0.4713	0.0278	21.747	3.1881	6.7010	8	2.53	15,844
BM25 ($\Gamma_{und}(u).\Gamma_{in}(v)$;$b$=0.1;$k$=1.0)	0.1395	0.4237	0.0292	23.209	3.3564	6.2102	8	2.34	7,151
ExtremeBM25 ($\Gamma_{und}(u).\Gamma_{in}(v)$;b=0.1)	0.1411	0.4220	0.0290	23.809	3.3788	6.2660	8	2.15	7,015
RA ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.1082	0.4182	0.0301	18.418	3.0887	5.8681	7	2.08	6,999
Pure Personalized PageRank (r=0.4)	0.0855	0.3621	0.0240	19.984	2.9912	6.0012	7	2.57	6,860
QLJM ($\Gamma_{und}(u).\Gamma_{in}(v).\lambda$=0.1)	0.1203	0.3873	0.0368	20.039	3.2067	6.1238	8	2.08	7,002
PropFlow ($\Gamma_{\text{out}}(u)$;l=5)	0.0967	0.3548	0.0233	20.975	3.1423	6.2463	8	2.02	6,739
Personalized HITS (Auth.; r=0.99)	0.0784	0.3502	0.0174	23.491	2.8391	6.1046	7	2.70	6,793
MatrixForest (α=0.001)	0.1085	0.3025	0.0349	18.186	3.3243	6.1785	9	2.26	6,851
Maximum Cosine Similarity	0.0809	0.2390	0.0230	31.139	3.2324	6.1779	9	2.38	8,908
Commute Time	0.0909	0.5882	0.0153	27.232	2.8526	6.6222	7	2.48	12,646

Table 27. Comparison of the different algorithms in terms of clustering coefficient. embeddedness and distance (1 Month interactions graph) (1 of 2)

	Clustering Coefficient		Embeddedness		Distances				
Algorithm	Global	Average	Global	Zero	ASL	Avg. Ecc.	Diameter	Rec. Dist	Edges ∞
Love (Hubs;k=50;α=0.3)	0.1411	0.4220	0.0159	26,313	2.7891	6.1162	7	2.21	7.080
Popularity	0.1082	0.4182	0.0154	24,186	2.6718	6.0008	7	2.31	6.919
Preferential Attachment ($\Gamma_{\text{in/out}}(u)$)	0.0855	0.3621	0.0154	24,186	2.6718	6.0008	7	2.02	6.739
Hitting Time	0.1203	0.3873	0.0152	27,312	2.8488	6.6222	7	2.73	6.860
Hub Promoted Index ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.0967	0.3548	0.0277	29,303	3.2275	6.4613	9	2.08	6.739
Hannon ($\Gamma_{in}(u)$)	0.0784	0.3502	0.0553	19,997	3.3959	9.1473	12	2.85	6.853
PageRank (r=0.1)	0.1085	0.3025	0.0149	27,723	2.7530	6.0192	7	2.69	6.795
Personalized PageRank (r=0.4)	0.0809	0.2390	0.0150	26,255	**2.6391**	5.9974	6	2.65	6.739
Jaccard ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.0909	0.5882	0.0496	19,265	3.3376	6.1225	9	**3.54**	16.917
Sorensen ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.0770	0.4232	0.05	19,265	3.34	6.12	9	2.39	14.128
Hub Depressed Index ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.0757	0.5647	0.0458	19,665	3.3387	6.1858	9	2.10	6.739
TF-IDF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.0757	0.5647	0.0490	19,161	3.2298	5.5666	8	2.45	15.635
Salton ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.0907	0.5894	0.05	20,630	3.31	6.00	8	2.73	6.860
Katz ($\Gamma_{out}(u)$;β=0.1)	0.0815	0.3551	0.0193	26,331	3.1920	6.4722	8	2.12	7.019
Distance (Dir.)	0.1090	0.3600	0.0258	17,376	3.2657	6.6212	9	2.32	7.223
LHN Index 1 ($\Gamma_{in}(u).\Gamma_{out}(v)$)	0.0850	0.5943	0.0283	34,379	3.3706	6.7880	9	2.73	6.860
PMF Sigmoid (k=50.0;λ=0.01;ϕ=0.01)	0.0737	0.5827	0.0154	25,940	3.0061	6.6209	7	2.42	6.987
LHN Index 2 (β=0.4)	0.1100	0.3660	0.0304	34,613	3.4982	7.0724	10	2.18	7.030
Random	0.11	0.37	**0.0148**	**103,656**	2.9488	**4.2654**	**5**	2.09	7.013
PMF Basic (k=40.0;λ=0.01;ϕ=0.01)	0.1041	0.3625	0.0153	47,006	3.0866	6.2671	7	2.34	6.871

Table 28. Comparison of the different algorithms in terms of clustering coefficient. embeddedness and distance (1 Month interactions graph) (2 of 2)

	Assortativity		Modularity			Weak ties		
Algorithm	Undirected	Directed (IN)	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
ImplicitMF (k=280.α=150;λ=40)	-0.1992	**-0.1247**	0.7084	0.7248	0.6808	18,686	19,290	29,629
UserBased kNN (k=120)	-0.1927	-0.1087	0.6720	0.6904	0.6514	20,304	19,119	31,373
Personalized SALSA (Auth.;r=0.99)	-0.2168	-0.1086	0.6927	0.7169	0.6770	24,906	24,730	35,022
Average Cosine Similarity	-0.2128	-0.1108	0.7149	0.7414	0.7098	20,947	20,281	29,795
ItemBased kNN (k=300)	-0.2142	-0.1127	0.7093	0.7366	0.7016	17,281	16,441	24,378
Centroid Cosine Similarity	-0.2316	-0.1136	0.7041	0.7415	0.6970	18,215	17,136	25,955
Local Path Index ($\Gamma_{\text{out}}(u)$;β=0.1;l=3)	-0.2612	-0.1202	0.6992	0.7438	0.6924	19,023	16,287	26,412
Adamic ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	-0.1566	-0.0895	0.7163	0.7280	0.7006	19,240	15,610	62,376
BIR ($\Gamma_{und}(u).\Gamma_{in}(v)$)	-0.1602	-0.0902	0.7121	0.7265	0.6966	17,479	18,712	26,305
Money (k=1000;α=0.3)	-0.2411	-0.1093	0.6810	0.7204	0.6710	18,283	18,991	27,011
FOAF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	-0.1655	-0.0919	0.7197	0.7359	0.7007	23,315	19,591	30,828
BM25 ($\Gamma_{und}(u).\Gamma_{in}(v)$;$b$=0.1;$k$=1.0)	-0.1776	-0.0881	0.7206	0.7469	0.7100	16,719	17,421	26,172
ExtremeBM25 ($\Gamma_{und}(u).\Gamma_{in}(v)$;b=0.1)	-0.1759	-0.0858	0.7207	0.7464	0.7102	16,894	15,776	24,634
RA ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	-0.1283	-0.0811	0.6929	0.7009	0.6777	16,936	15,882	24,627
Pure Personalized PageRank (r=0.4)	-0.1620	-0.0916	0.6538	0.6818	0.6451	21,952	23,216	30,779
QLJM ($\Gamma_{und}(u).\Gamma_{in}(v).\lambda$=0.1)	-0.1188	-0.0771	0.7183	0.7379	0.7033	29,598	26,619	36,730
PropFlow ($\Gamma_{\text{out}}(u)$;l=5)	-0.1824	-0.1023	0.6515	0.6822	0.6418	17,525	17,381	26,217
Personalized HITS (Auth.; r=0.99)	-0.2393	-0.1043	0.5570	0.5904	0.5523	29,989	26,507	37,363
MatrixForest (α=0.001)	-0.1258	-0.0949	0.6964	0.7134	0.6804	42,791	38,897	49,969
Maximum Cosine Similarity	-0.1735	-0.1016	0.6872	0.7172	0.6703	21,698	21,426	30,563
Commute Time	-0.2652	-0.0967	0.4262	0.4551	0.4284	22,544	20,519	32,011

Table 29. Comparison of the different algorithms in terms of assortativity. modularity and weak ties (1 Month interactions graph) (1 of 2)

Algorithm	Assortativity		Modularity			Weak ties		
	Undirected	Directed (IN)	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
Love (Hubs;k=50;α=0.3)	-0.2554	-0.1040	0.4782	0.4773	0.4769	57,136	59,018	26,432
Popularity	**-0.2751**	-0.1088	0.4253	**0.4299**	0.4232	64,888	64,299	71,393
Preferential Attachment ($\Gamma_{\text{in/out}}(u)$)	**-0.2751**	-0.1088	0.4253	**0.4299**	0.4232	64,888	64,299	71,393
Hitting Time	-0.2674	-0.0972	0.4205	0.4493	0.4226	72,491	63,185	75,656
Hub Promoted Index ($\Gamma_{in}(u).\Gamma_{und}(v)$)	-0.1409	-0.1017	0.6870	0.6958	0.6650	22,478	24,014	33,091
Hannon ($\Gamma_{in}(u)$)	-0.0686	-0.0623	0.7245	0.7434	0.7060	16,190	16,376	25,587
PageRank (r=0.1)	-0.2700	-0.1009	**0.4116**	0.4333	**0.4104**	74,604	68,237	80,274
Personalized PageRank (r=0.4)	-0.2747	-0.1117	0.4263	0.4372	0.4153	71,641	67,720	80,713
Jaccard ($\Gamma_{und}(u).\Gamma_{in}(v)$)	-0.0821	-0.0710	0.7177	0.7341	0.6924	17,652	18,041	28,604
Sorensen ($\Gamma_{und}(u).\Gamma_{in}(v)$)	-0.0821	-0.0710	0.7177	0.7341	0.6924	17,652	18,041	28,604
Hub Depressed Index ($\Gamma_{und}(u).\Gamma_{in}(v)$)	-0.0788	-0.0712	0.7216	0.7339	0.6953	16,916	18,048	28,037
TF-IDF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	-0.0679	-0.0624	0.6971	0.7109	0.6754	21,753	21,945	32,122
Salton ($\Gamma_{und}(u).\Gamma_{in}(v)$)	-0.0912	-0.0732	0.7035	0.7224	0.6809	20,221	20,002	30,769
Katz ($\Gamma_{out}(u)$;β=0.1)	-0.2282	-0.1115	0.4924	0.5301	0.4832	59,935	51,034	68,682
Distance (Dir.)	-0.0687	-0.0657	0.6573	0.6871	0.6330	28,403	25,711	38,951
LHN Index 1 ($\Gamma_{in}(u).\Gamma_{out}(v)$)	-0.1498	-0.0955	0.7310	0.7514	0.7167	14,827	15,087	23,671
PMF Sigmoid (k=50.0;λ=0.01;ϕ=0.01)	-0.2620	-0.0866	0.4498	0.4632	0.4364	65,848	61,670	75,383
LHN Index 2 (β=0.4)	-0.0731	-0.0720	0.6784	0.7081	0.6636	24,363	22,143	33,762
Random	-0.0508	-0.0553	0.4540	0.4682	0.4435	65,015	61,392	73,976
PMF Basic (k=40.0;λ=0.01;ϕ=0.01)	-0.1343	-0.0678	0.4602	0.4688	0.4493	63,316	62,202	72,625

Table 30. Comparison of the different algorithms in terms of assortativity. modularity and weak ties (1 Month interactions graph) (2 of 2)

Algorithm	Comm-PairGini			Comm-InGini		
	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
ImplicitMF (k=280.α=150;λ=40)	0.1902	0.8270	0.0151	0.3817	0.9536	0.0805
UserBased kNN (k=120)	0.1805	0.7812	0.0140	0.3609	0.9101	0.0740
Personalized SALSA (Auth.;r=0.99)	0.1761	0.7847	0.0127	0.3501	0.8597	0.0706
Average Cosine Similarity	0.1807	0.7746	0.0141	0.3639	0.8817	0.0829
ItemBased kNN (k=300)	0.1775	0.7491	0.0141	0.3497	0.9026	0.0798
Centroid Cosine Similarity	0.1774	0.7175	0.0129	0.3509	0.8749	0.0726
Local Path Index ($\Gamma_{\text{out}}(u)$;β=0.1;l=3)	0.1549	0.6442	0.0063	0.3110	0.7597	0.0374
Adamic ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.1824	0.8079	0.0150	0.3804	0.9016	0.0903
BIR ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.1822	0.7981	0.0151	0.3843	0.8895	0.0902
Money (k=1000;α=0.3)	0.1726	0.7492	0.0125	0.3559	0.8755	0.0716
FOAF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.1833	0.7845	0.0149	0.3709	0.9276	0.0868
BM25 ($\Gamma_{und}(u).\Gamma_{in}(v)$;$b$=0.1;$k$=1.0)	0.1840	0.7992	0.0162	0.3935	0.9270	0.0964
ExtremeBM25 ($\Gamma_{und}(u).\Gamma_{in}(v)$;b=0.1)	0.1849	0.8019	0.0165	0.3975	0.9308	0.0979
RA ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.1856	0.8128	0.0158	0.4034	0.8738	0.1016
Pure Personalized PageRank (r=0.4)	0.1924	0.8099	0.0148	0.4361	0.9113	0.0868
QLJM ($\Gamma_{und}(u).\Gamma_{in}(v).\lambda$=0.1)	0.1868	0.7968	0.0171	0.4067	0.9013	0.1037
PropFlow ($\Gamma_{\text{out}}(u)$;l=5)	0.1895	0.7791	0.0150	0.4228	0.8529	0.0858
Personalized HITS (Auth.; r=0.99)	0.1347	0.6075	0.0077	0.2186	0.5441	0.0428
MatrixForest (α=0.001)	0.1883	0.7713	0.0170	0.4089	0.8251	0.1011
Maximum Cosine Similarity	0.1923	0.8222	0.0145	0.3913	0.9446	0.0941
Commute Time	0.1640	0.6717	0.0068	0.3205	0.6555	0.0374

Table 31. Comparison of the different algorithms in terms of community Gini (1 Month interactions graph) (1 of 2)

Algorithm	Comm-PairGini			Comm-InGini		
	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
Love (Hubs;k=50;α=0.3)	0.1167	0.4816	0.0102	0.1883	0.4301	0.0635
Popularity	0.1166	0.5876	0.0057	0.2180	0.6186	0.0352
Preferential Attachment ($\Gamma_{\text{in/out}}(u)$)	0.1166	0.5876	0.0057	0.2180	0.6186	0.0352
Hitting Time	0.1675	0.6892	0.0069	0.3235	0.6692	0.0379
Hub Promoted Index ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.1847	0.8407	0.0148	0.3714	0.9522	0.0903
Hannon ($\Gamma_{in}(u)$)	0.1969	0.8085	0.0176	0.4086	0.9421	0.1090
PageRank (r=0.1)	0.1946	0.8179	0.0080	0.4089	0.8768	0.0446
Personalized PageRank (r=0.4)	0.1673	0.8427	0.0080	0.3694	0.9620	0.0458
Jaccard ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.2010	0.8196	0.0186	0.4306	0.9230	0.1199
Sorensen ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.2010	0.8196	0.0186	0.4306	0.9230	0.1199
Hub Depressed Index ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.1980	0.8010	**0.0191**	0.4238	0.9105	0.1198
TF-IDF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.1995	0.8235	0.0189	**0.4536**	0.8763	**0.1383**
Salton ($\Gamma_{und}(u).\Gamma_{in}(v)$)	**0.2028**	0.8195	0.0184	0.4353	0.9170	0.1213
Katz ($\Gamma_{out}(u)$;β=0.1)	0.1676	0.6456	0.0089	0.3629	0.6667	0.0499
Distance (Dir.)	0.1996	0.8291	0.0174	0.4250	0.9456	0.1245
LHN Index 1 ($\Gamma_{in}(u).\Gamma_{out}(v)$)	0.1832	0.7794	0.0158	0.3706	0.8606	0.1027
PMF Sigmoid (k=50.0;λ=0.01;ϕ=0.01)	0.1524	0.8152	0.0078	0.3157	0.8367	0.0437
LHN Index 2 (β=0.4)	0.1921	0.8418	0.0164	0.3972	**0.9714**	0.1092
Random	0.1913	**0.8428**	0.0188	0.4020	0.9157	0.1259
PMF Basic (k=40.0;λ=0.01;ϕ=0.01)	0.1742	0.8149	0.0129	0.3410	0.8414	0.0721

Table 32. Comparison of the different algorithms in terms of community Gini (1 Month interactions graph) (2 of 2)

Follows

In this section. we show the complete results for the follows graph included in this dataset. We divide the results in 9 different tables.

First, in Table 33. we show the values of the accuracy metrics. Then. Tables 34 and 35 show the values for the novelty and diversity metrics. Tables 36 and 37 display the values for embeddedness. clustering coefficient and distance metrics. Tables 38 and 39 show the modularity of the extended graphs. the number of weak ties and the degree assortativity. Finally. Tables 40 and 41 show the community Gini coefficients.

Algorithm	P@10	R@10	nDCG@10	Algorithm	P@10	R@10	nDCG@10
ExtremeBM25 ($\Gamma_{und}(u).\Gamma_{und}(v)$;$b$=0.99)	**0.00674**	**0.03273**	**0.02633**	Commute Time	0.00068	0.00291	0.00134
BM25 ($\Gamma_{und}(u).\Gamma_{und}(v)$;$b$=1.0;$k$=1000.0)	**0.00674**	**0.03273**	**0.02633**	PropFlow ($\Gamma_{out}(u)$;l=5)	0.00061	0.00341	0.00209
PageRank (r=0.6)	0.00321	0.01211	0.00944	Hannon ($\Gamma_{in}(u)$)	0.00049	0.00156	0.00103
Personalized PageRank (r=0.5)	0.00300	0.01034	0.00778	TF-IDF ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.00043	0.00102	0.00072
Popularity	0.00279	0.00924	0.00835	Salton ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.00042	0.00115	0.00087
PrefAttach ($\Gamma_{in}(u)$)	0.00279	0.00924	0.00835	Centroid Cosine Similarity	0.00039	0.00207	0.00109
Local Path Index ($\Gamma_{und}(u)$;β=0.3;l=5)	0.00274	0.00664	0.00552	Hitting Time	0.00039	0.00121	0.00059
Katz ($\Gamma_{out}(u)$;β=0.9)	0.00199	0.00967	0.00670	Item-Based kNN (k=240)	0.00039	0.00196	0.00124
Pure Personalized PageRank (r=0.5)	0.00194	0.00354	0.00356	Jaccard ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.00039	0.00115	0.00087
Res. Allocation ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.00169	0.00437	0.00389	Sorensen ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.00039	0.00115	0.00087
Adamic ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.00151	0.00315	0.00309	Hub Depressed Index ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.00034	0.00110	0.00081
FOAF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.00148	0.00309	0.00294	Average Cosine Similarity	0.00029	0.00165	0.00091
BIR ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.00147	0.00282	0.00289	Matrix Forest (α=0.01)	0.00020	0.00099	0.00058
Love (Auth.;k=100;α=0.1)	0.00144	0.00875	0.00401	Distance (Undir.)	0.00011	0.00010	0.00016
Personalized HITS (Auth.;α=0.1)	0.00139	0.00872	0.00450	LHN Index 1 ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.00011	0.00022	0.00019
Personalized SALSA (Auth.;α=0.1)	0.00139	0.00872	0.00616	LHN Index 2 (β=0.1)	0.00009	0.00014	0.00013
Money (Hubs;k=10;α=0.3)	0.00120	0.00046	0.00122	PMF Sigmoid (k=50;λ=0.01;ϕ=0.01)	0.00004	0.00001	0.00003
ImplicitMF (k=30;λ=150.0;α=40.0)	0.00114	0.00630	0.00376	Maximum Cosine Similarity	0.00002	0.00014	0.00015
HubPromotedIndex ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.00111	0.00190	0.00188	PMF Basic (k=50;λ=0.01;ϕ=0.01)	0.00002	0.00020	0.00020
QLJM ($\Gamma_{in}(u).\Gamma_{in}(v)$;$\lambda$=0.1)	0.00083	0.00187	0.00151	Random	0.00002	0.00003	0.00004
User-Based kNN (k=70)	0.00075	0.00446	0.00232				

Table 33. Comparison of the different algorithms in terms of P@10. R@10 and nDCG@10 (1 Month follows graph)

Algorithm	Novelty		Diversity		ERR-IA			Subtopic-Recall		
	PC	PD	ILD	Gini	Infomap	Louvain	Leading Vector	Infomap	Louvain	Leading Vector
ExtremeBM25 ($\Gamma_{und}(u).\Gamma_{und}(v)$;$b$=0.99)	0.9469	0.4474	0.5293	0.0086	**0.0089**	**0.0091**	**0.0127**	**0.0013**	**0.0161**	**0.0318**
BM25 ($\Gamma_{und}(u).\Gamma_{und}(v)$;$b$=1.0;$k$=1000.0)	0.9469	0.4473	0.5291	0.0087	**0.0089**	**0.0091**	**0.0127**	**0.0013**	**0.0161**	**0.0318**
PageRank (r=0.6)	0.7439	0.7608	0.5564	0.0018	0.0018	0.0026	0.0031	0.0005	0.0060	0.0116
Personalized PageRank (r=0.5)	0.7411	0.7633	0.5373	0.0018	0.0016	0.0021	0.0025	0.0004	0.0052	0.0101
Popularity	0.7301	**0.7867**	0.5039	0.0018	0.0015	0.0024	0.0028	0.0004	0.0047	0.0090
PrefAttach ($\Gamma_{in}(u)$)	0.7301	**0.7867**	0.5039	0.0018	0.0015	0.0024	0.0028	0.0004	0.0047	0.0090
Local Path Index ($\Gamma_{und}(u)$;β=0.3;l=5)	0.8328	0.6202	0.1618	0.0046	0.0016	0.0006	0.0027	0.0003	0.0036	0.0066
Katz ($\Gamma_{out}(u)$;β=0.9)	0.9668	0.6417	0.4823	0.0053	0.0014	0.0019	0.0025	0.0004	0.0049	0.0097
Pure Personalized PageRank (r=0.5)	0.8695	0.3436	0.3257	0.0214	0.0017	0.0015	0.0022	0.0002	0.0026	0.0048
Res. Allocation ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.8848	0.3371	0.2669	0.0698	0.0012	0.0013	0.0020	0.0002	0.0025	0.0048
Adamic ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.8598	0.3183	0.2457	0.0314	0.0010	0.0012	0.0018	0.0002	0.0020	0.0039
FOAF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.8563	0.3213	0.2451	0.0268	0.0010	0.0011	0.0017	0.0002	0.0019	0.0038
BIR ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.8570	0.3207	0.2504	0.0274	0.0010	0.0012	0.0017	0.0002	0.0019	0.0037
Love (Auth.;k=100;α=0.1)	0.7742	0.4665	0.3045	0.0017	0.0002	0.0019	0.0010	0.0003	0.0035	0.0071
Personalized HITS (Auth.;α=0.1)	0.7420	0.7677	0.2836	0.0022	0.0001	0.0007	0.0006	0.0003	0.0034	0.0069
Personalized SALSA (Auth.;α=0.1)	0.7431	0.6306	0.5578	0.0027	0.0002	0.0012	0.0016	0.0003	0.0034	0.0069
Money (Hubs;k=10;α=0.3)	0.8265	0.7505	0.6143	0.0015	0.0008	0.0007	0.0009	0.0001	0.0011	0.0018
ImplicitMF (k=30;λ=150.0;α=40.0)	0.8904	0.3102	0.2934	0.0494	0.0001	0.0005	0.0014	0.0001	0.0009	0.0057
HubPromotedIndex ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.9385	0.3670	0.2527	0.1192	0.0006	0.0005	0.0012	0.0001	0.0007	0.0026
QLJM ($\Gamma_{in}(u).\Gamma_{in}(v)$;$\lambda$=0.1)	0.9552	0.3579	0.2432	0.2564	0.0008	0.0007	0.0009	0.0001	0.0013	0.0026

Table 34. Comparison of the different algorithms in terms novelty and diversity (1 Month follows graph) (1 of 2)

	Novelty		Diversity		ERR-IA			Subtopic Recall		
Algorithm	PC	PD	ILD	Gini	Infomap	Louvain	Leading Vector	Infomap	Louvain	Leading Vector
User-Based kNN (k=70)	0.8617	0.2988	0.2339	0.0282	0.0002	0.0006	0.0008	0.0002	0.0019	0.0037
Commute Time	0.9306	0.6391	0.2001	0.0012	0.0003	0.0003	0.0004	0.0001	0.0015	0.0031
PropFlow ($\Gamma_{out}(u)$;l=5)	0.8687	0.3489	0.3296	0.0175	0.0003	0.0004	0.0009	0.0001	0.0015	0.0030
Hannon ($\Gamma_{in}(u)$)	0.9778	0.3237	0.2071	0.4240	0.0006	0.0005	0.0007	0.0001	0.0009	0.0018
TF-IDF ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.9863	0.3710	0.2693	0.3478	0.0005	0.0004	0.0005	0.0001	0.0008	0.0015
Salton ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.9853	0.3644	0.2439	0.2455	0.0006	0.0006	0.0006	0.0001	0.0008	0.0016
Centroid Cosine Similarity	0.8451	0.3041	0.1244	0.0202	0.0001	0.0004	0.0004	0.0001	0.0009	0.0019
Hitting Time	0.9369	0.6394	0.1999	0.0012	0.0001	0.0002	0.0002	0.0001	0.0008	0.0016
Item-Based kNN (k=240)	0.8669	0.2945	0.1519	0.0306	0.0009	0.0008	0.0005	0.0001	0.0013	0.0019
Jaccard ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.9884	0.3651	0.2476	0.2502	0.0006	0.0006	0.0007	0.0001	0.0008	0.0016
Sorensen ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.9884	0.3651	0.2476	0.2502	0.0006	0.0006	0.0007	0.0001	0.0008	0.0016
Hub Depressed Index ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.9899	0.3678	0.2555	0.2581	0.0010	0.0012	0.0006	0.0002	0.0028	0.0014
Average Cosine Similarity	0.8639	0.2815	0.1219	0.0331	0.0001	0.0003	0.0003	0.0001	0.0007	0.0015
Matrix Forest (α=0.01)	0.9898	0.2957	0.1766	0.2746	0.0002	0.0002	0.0003	0.0000	0.0005	0.0010
Distance (Undir.)	0.9914	0.5708	0.6234	0.8565	0.0001	0.0001	0.0001	0.0000	0.0003	0.0005
LHN Index 1 ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.9956	0.4102	0.3233	0.1125	0.0002	0.0001	0.0002	0.0000	0.0003	0.0005
LHN Index 2 (β=0.1)	**0.9998**	0.3448	0.1900	0.0380	0.0001	0.0001	0.0002	0.0000	0.0002	0.0003
PMF Sigmoid (k=50;λ=0.01;ϕ=0.01)	0.9877	0.6569	0.4782	0.0011	0.0000	0.0000	0.0000	0.0000	0.0001	0.0002
Maximum Cosine Similarity	0.8610	0.3675	0.1776	0.0291	0.0000	0.0000	0.0001	0.0000	0.0001	0.0001
PMF Basic (k=50;λ=0.01;ϕ=0.01)	0.9946	0.6617	**0.6670**	0.0160	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001
Random	0.9939	0.6588	0.6661	**0.9713**	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001

Table 35. Comparison of the different algorithms in terms novelty and diversity (1 Month follows graph) (2 of 2)

	Clustering Coefficient		Embeddedness		Distances				
Algorithm	Global	Average	Global	Zero	ASL	Avg. Ecc.	Diameter	Rec. Dist	Edges ∞
ExtremeBM25 ($\Gamma_{und}(u).\Gamma_{und}(v)$;$b$=0.99)	0.2639	0.3365	0.0452	18,526	3.2886	9.8545	19	2.1161	4,098
BM25 ($\Gamma_{und}(u).\Gamma_{und}(v)$;$b$=1.0;$k$=1000.0)	0.2755	0.3658	0.0453	18,462	3.2579	9.8234	13	2.1160	4,082
PageRank (r=0.6)	0.2777	0.4132	0.0439	16,628	3.0523	6.3017	7	2.1779	1,810
Personalized PageRank (r=0.5)	0.2770	0.4131	0.0440	16,610	3.0512	6.3065	7	2.1717	1,810
Popularity	0.2743	0.4130	0.0439	16,728	3.0416	6.2757	7	2.1923	1,810
PrefAttach ($\Gamma_{in}(u)$)	0.2743	0.4130	0.0439	16,728	3.0416	6.2757	7	2.1923	1,810
Local Path Index ($\Gamma_{und}(u)$;β=0.3;l=5)	0.2658	0.3934	0.0463	15,961	3.0065	6.1943	7	2.0863	1,810
Katz ($\Gamma_{out}(u)$;β=0.9)	0.2293	0.2721	0.0435	19,923	3.1286	6.3486	8	2.6466	1,810
Pure Personalized PageRank (r=0.5)	0.2607	0.3424	0.0486	14,024	3.0610	6.2477	8	2.0054	1,810
Res. Allocation ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.2584	0.3750	0.0521	12,154	3.0042	6.0439	7	2.0262	1,843
Adamic ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.2647	0.3702	0.0508	14,319	3.0248	6.2063	8	2.0206	1,843
FOAF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.2649	0.3651	0.0504	14,695	3.0254	6.2229	8	2.0205	1,842
BIR ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.2667	0.3635	0.0503	14,586	3.0301	6.2160	8	2.0187	1,842
Love (Auth.;k=100;α=0.1)	0.2694	0.4064	0.0455	15,973	2.9036	6.1024	7	2.1471	1,810
Personalized HITS (Auth.;α=0.1)	0.2781	0.3950	0.0439	18,618	3.0424	6.1313	7	2.2042	1,810
Personalized SALSA (Auth.;α=0.1)	0.2735	0.3905	0.0445	18,112	3.0013	6.1171	7	2.1212	1,810
Money (Hubs;k=10;α=0.3)	0.2694	0.4064	0.0441	16,123	3.0163	6.2505	7	2.3940	20,569
ImplicitMF (k=30;λ=150.0;α=40.0)	0.2376	0.2924	0.0520	16,276	2.9608	6.0341	8	2.2313	11,883
HubPromotedIndex ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.2247	0.3446	0.0523	16,133	3.0738	5.8412	8	2.3698	14,113
QLJM ($\Gamma_{in}(u).\Gamma_{in}(v)$;$\lambda$=0.1)	0.2512	0.3389	0.0584	11,156	2.9962	5.9996	7	2.2067	9,336

Table 36. Comparison of the different algorithms in terms of clustering coefficient. embeddedness and distance (1 Month follows graph) (1 of 2)

	Clustering Coefficient		Embeddedness		Distances				
Algorithm	Global	Average	Global	Zero	ASL	Avg. Ecc.	Diameter	Rec. Dist	Edges ∞
User-Based kNN (k=70)	0.2548	0.3218	0,0509	15,727	3.0238	6.2227	8	2.0642	2,744
Commute Time	0.2722	0.4046	0,0440	17,409	3.0828	6.2898	8	2.2764	1,810
PropFlow ($\Gamma_{out}(u)$;l=5)	0.2611	0.3392	0,0482	15,972	3.0552	6.2684	8	2.0099	1,810
Hannon ($\Gamma_{in}(u)$)	0.2432	0.2915	0,0615	13,896	3.1005	10.2943	12	2.2493	6,940
TF-IDF ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.2432	0.2815	0,0596	16,728	3.0699	5.7569	8	2.3489	14,942
Salton ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.2446	0.2872	0,0586	16,424	3.0804	6.0778	8	2.3355	13,311
Centroid Cosine Similarity	0.2601	0.3245	0,0504	17,211	3.0338	6.3111	8	2.0382	1,820
Hitting Time	0.2699	0.4052	0,0440	17,401	3.0813	6.2897	7	2.3067	1,810
Item-Based kNN (k=240)	0.2487	0.3036	0,0514	15,498	3.0302	6.3378	8	2.0565	2,885
Jaccard ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.2417	0.2788	0,0582	16,535	3.0808	6.0582	8	2.3523	13,561
Sorensen ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.2417	0.2788	0,0582	16,535	3.0808	6.0582	8	2.3523	13,561
Hub Depressed Index ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.2374	0.2713	0,0570	17,581	3.0161	6.1988	8	2.0705	2,614
Average Cosine Similarity	0.2558	0.3075	0,0533	15,935	3.0394	6.6307	8	2.0381	1,820
Matrix Forest (α=0.01)	0.2324	0.2526	0,0616	10,136	3.0302	5.8740	8	2.0029	1,810
Distance (Undir.)	**0.1963**	0.1756	0,0432	78,259	2.8283	5.0471	6	2.9737	15,056
LHN Index 1 ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.2161	0.2588	0,0498	19,648	3.0439	6.0400	8	2.5130	20,041
LHN Index 2 (β=0.1)	0.2345	0.2716	0,0479	23,079	3.1183	6.7234	8	2.6580	**76,040**
PMF Sigmoid (k=50;λ=0.01;ϕ=0.01)	0.2652	0.4018	0,0444	16,072	2.9360	6.1826	7	2.9820	1,810
Maximum Cosine Similarity	0.2466	0.2925	0,0481	17,731	3.0173	6.5958	8	2.0673	1,826
PMF Basic (k=50;λ=0.01;ϕ=0.01)	0.2053	0.2147	0,0437	30,902	2.8895	6.0053	7	3.1285	1,900
Random	0.1970	**0.1505**	**0,0427**	**89,092**	**2.7989**	**4.0463**	**5**	**3.1553**	17,315

Table 37. Comparison of the different algorithms in terms of clustering coefficient. embeddedness and distance (1 Month follows graph) (2 of 2)

Algorithm	Assortativity		Modularity			Weak ties		
	Undirected	Directed (IN)	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
ExtremeBM25 ($\Gamma_{und}(u).\Gamma_{und}(v)$;$b$=0.99)	-0.1889	0.0762	0.5950	0.7058	0.6186	59,610	24,768	42,441
BM25 ($\Gamma_{und}(u).\Gamma_{und}(v)$;$b$=1.0;$k$=1000.0)	-0.1889	0.0762	0.5951	0.7058	0.6186	59,577	24,769	42,432
PageRank (r=0.6)	-0.1786	-0.0700	0.5167	0.6199	0.5583	97,951	55,366	66,749
Personalized PageRank (r=0.5)	-0.1785	-0.0700	0.5168	0.6271	0.5633	97,450	52,695	64,116
Popularity	-0.1801	-0.0700	**0.5122**	**0.6049**	0.5466	98,063	60,973	72,155
PrefAttach ($\Gamma_{in}(u)$)	-0.1801	-0.0700	**0.5122**	**0.6049**	0.5466	98,063	60,973	72,155
Local Path Index ($\Gamma_{und}(u)$;β=0.3;l=5)	-0.1800	-0.0686	0.5872	0.7370	0.6353	60,131	13,253	32,082
Katz ($\Gamma_{out}(u)$;β=0.9)	-0.3016	-0.1200	0.5374	0.6375	0.5559	89,855	49,570	74,517
Pure Personalized PageRank (r=0.5)	-0.2851	-0.1000	0.6167	0.7402	0.6524	47,245	12,151	25,557
Res. Allocation ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	-0.2755	-0.0900	0.6229	0.7346	0.6558	43,861	14,292	24,734
Adamic ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	-0.2900	-0.0982	0.6259	0.7498	0.6642	41,858	8,744	20,154
FOAF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	-0.2869	-0.1000	0.6238	0.7520	0.6627	42,733	7,938	20,655
BIR ($\Gamma_{und}(u).\Gamma_{in}(v)$)	-0.2900	-0.0978	0.6251	0.7518	0.6639	42,275	7,990	20,131
Love (Auth.;k=100;α=0.1)	-0.2329	-0.0900	0.5418	0.6292	0.5598	81,256	53,952	69,800
Personalized HITS (Auth.;α=0.1)	-0.1789	-0.0700	0.5202	0.6267	0.5629	93,907	52,697	64,074
Personalized SALSA (Auth.;α=0.1)	-0.1893	-0.0800	0.5472	0.6629	0.5872	80,949	39,968	53,862
Money (Hubs;k=10;α=0.3)	-0.1701	-0.0700	0.5182	0.6062	0.5474	94,367	60,972	72,657
ImplicitMF (k=30;λ=150.0;α=40.0)	**-0.3112**	**-0.1300**	0.6290	0.7535	0.6666	39,776	7,368	29,036
HubPromotedIndex ($\Gamma_{in}(u).\Gamma_{und}(v)$)	-0.2796	-0.1200	0.6174	0.7259	0.6468	45,955	17,509	31,526
QLJM ($\Gamma_{in}(u).\Gamma_{in}(v)$;$\lambda$=0.1)	-0.2545	-0.0900	0.6202	0.7257	0.6491	44,819	17,558	28,172

Table 38. Comparison of the different algorithms in terms of assortativity. modularity and weak ties (1 Month follows graph) (1 of 2)

Algorithm	Assortativity		Modularity			Weak ties		
	Undirected	Directed (IN)	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
User-Based kNN (k=70)	-0.2980	-0.1200	0.6327	0.7454	0.6636	38,937	10,267	20,418
Commute Time	-0.1715	-0.0800	0.5386	0.6417	0.5469	90,731	47,615	83,134
PropFlow ($\Gamma_{out}(u)$;l=5)	-0.2856	-0.1000	0.6139	0.7410	0.6516	48,655	11,882	25,973
Hannon ($\Gamma_{in}(u)$)	-0.2300	-0.0824	0.6295	0.7367	0.6565	40,930	13,486	24,828
TF-IDF ($\Gamma_{in}(u).\Gamma_{und}(v)$)	-0.2368	-0.0900	0.6185	0.7205	0.6456	46,327	19,490	30,612
Salton ($\Gamma_{in}(u).\Gamma_{und}(v)$)	-0.2433	-0.0900	0.6203	0.7242	0.6460	45,365	18,132	30,585
Centroid Cosine Similarity	-0.2916	-0.1100	0.6245	0.7585	0.6638	41,798	5,482	19,799
Hitting Time	-0.1708	-0.0800	0.5376	0.6414	**0.5438**	90,909	47,732	85,832
Item-Based kNN (k=240)	-0.3011	-0.1200	0.6276	0.7355	0.6531	42,247	13,958	18,897
Jaccard ($\Gamma_{in}(u).\Gamma_{und}(v)$)	-0.2395	-0.0900	0.6199	0.7235	0.6452	45,578	18,379	30,954
Sorensen ($\Gamma_{in}(u).\Gamma_{und}(v)$)	-0.2395	-0.0900	0.6199	0.7235	0.6452	45,578	18,379	30,954
Hub Depressed Index ($\Gamma_{in}(u).\Gamma_{und}(v)$)	-0.2400	-0.0940	0.6188	0.7226	0.6440	46,177	18,705	25,463
Average Cosine Similarity	-0.2963	-0.1200	0.6325	0.7578	0.6720	38,030	5,752	16,470
Matrix Forest (α=0.01)	-0.2310	-0.1000	0.6340	0.7307	0.6573	38,569	15,772	25,707
Distance (Undir.)	-0.2141	-0.0900	0.5598	0.6732	0.5879	75,664	36,515	55,869
LHN Index 1 ($\Gamma_{in}(u).\Gamma_{und}(v)$)	-0.2580	-0.1200	0.6069	0.7167	0.6313	52,602	20,893	37,804
LHN Index 2 (β=0.1)	-0.2662	-0.1100	0.6237	0.7436	0.6548	44,467	10,946	25,915
PMF Sigmoid (k=50;λ=0.01;ϕ=0.01)	-0.1683	-0.0700	0.5395	0.6317	0.5627	81,847	52,265	67,126
Maximum Cosine Similarity	-0.2660	-0.1000	0.6073	0.7328	0.6461	50,523	14,765	28,639
PMF Basic (k=50;λ=0.01;ϕ=0.01)	-0.2926	-0.1200	0.5397	0.6387	0.5638	84,772	49,076	67,776
Random	-0.2101	-0.0900	0.5387	0.6377	0.5630	86,474	49,491	68,252

Table 39. Comparison of the different algorithms in terms of assortativity. modularity and weak ties (1 Month follows graph) (2 of 2)

Algorithm	Comm-PairGini			Comm-InGini		
	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
ExtremeBM25 ($\Gamma_{und}(u).\Gamma_{und}(v)$;$b$=0.99)	0.7244	0.8434	0.0145	0.8807	0.8434	0.0814
BM25 ($\Gamma_{und}(u).\Gamma_{und}(v)$;$b$=1.0;$k$=1000.0)	0.7224	0.8435	0.0145	0.8807	0.8435	0.0814
PageRank (r=0.6)	0.7229	0.9371	0.0103	0.8846	0.9371	0.0619
Personalized PageRank (r=0.5)	0.7083	0.9150	0.0104	0.8447	0.9150	0.0611
Popularity	0.7022	0.9903	0.0100	0.8177	0.9903	0.0603
PrefAttach ($\Gamma_{in}(u)$)	0.7022	0.9903	0.0100	0.8177	0.9903	0.0603
Local Path Index ($\Gamma_{und}(u)$;β=0.3;l=5)	0.6925	0.9172	0.0120	0.8794	0.9172	0.0686
Katz ($\Gamma_{out}(u)$;β=0.9)	0.6817	0.8871	0.0131	0.7521	0.8871	0.0782
Pure Personalized PageRank (r=0.5)	0.6894	0.8854	0.0128	0.8224	0.8854	0.0719
Res. Allocation ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.6900	0.8195	0.0132	0.8511	0.8195	0.0744
Adamic ($\Gamma_{und}(u).\Gamma_{in}(v).\Gamma_{und}(w)$)	0.6798	0.8255	0.0127	0.8399	0.8255	0.0692
FOAF ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.6787	0.8237	0.0127	0.8370	0.8237	0.0684
BIR ($\Gamma_{und}(u).\Gamma_{in}(v)$)	0.6804	0.8367	0.0127	0.8412	0.8367	0.0690
Love (Auth.;k=100;α=0.1)	0.5881	0.5066	0.0102	0.6397	0.5066	0.0540
Personalized HITS (Auth.;α=0.1)	0.6700	0.8799	0.0103	0.7715	0.8799	0.0604
Personalized SALSA (Auth.;α=0.1)	0.7178	**0.9910**	0.0110	0.8580	**0.9910**	0.0647
Money (Hubs;k=10;α=0.3)	0.7013	0.8960	0.0097	0.8176	0.8960	0.0593
ImplicitMF (k=30;λ=150.0;α=40.0)	0.6972	0.8398	0.0132	0.8544	0.8398	0.0725
HubPromotedIndex ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.7069	0.8086	0.0153	0.8513	0.8086	0.0871
QLJM ($\Gamma_{in}(u).\Gamma_{in}(v)$;$\lambda$=0.1)	0.7082	0.8058	0.0134	0.8514	0.8058	0.0740

Table 40. Comparison of the different algorithms in terms of community Gini (1 Month follows graph) (1 of 2)

Algorithm	Comm-PairGini			Comm-InGini		
	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
User-Based kNN (k=70)	0.6991	0.8596	0.0129	0.8687	0.8596	0.0708
Commute Time	0.6089	0.9822	0.0113	0.6299	0.9822	0.0714
PropFlow ($\Gamma_{out}(u)$;l=5)	0.6831	0.8686	0.0128	0.8153	0.8686	0.0713
Hannon ($\Gamma_{in}(u)$)	0.7123	0.8491	0.0147	0.8780	0.8491	0.0805
TF-IDF ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.7150	0.8021	0.0140	0.8693	0.8021	0.0805
Salton ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.7147	0.7925	0.0149	0.8687	0.7925	0.0848
Centroid Cosine Similarity	0.6951	0.8714	0.0126	0.8693	0.8714	0.0681
Hitting Time	0.5874	0.9826	0.0112	0.6012	0.9826	0.0704
Item-Based kNN (k=240)	0.6938	0.8273	0.0127	0.8590	0.8273	0.0687
Jaccard ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.7145	0.7946	0.0151	0.8674	0.7946	0.0861
Sorensen ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.7145	0.7946	0.0151	0.8674	0.7946	0.0861
Hub Depressed Index ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.7144	0.7983	0.0129	0.8676	0.7983	0.0706
Average Cosine Similarity	0.6983	0.8676	0.0128	0.8629	0.8676	0.0697
Matrix Forest (α=0.01)	0.6860	0.7840	0.0142	0.8239	0.7840	0.0840
Distance (Undir.)	0.7254	0.8949	0.0138	0.8707	0.8949	0.0782
LHN Index 1 ($\Gamma_{in}(u).\Gamma_{und}(v)$)	0.7233	0.7956	**0.0156**	0.8839	0.7956	**0.0905**
LHN Index 2 (β=0.1)	0.7089	0.8655	0.0145	**0.9475**	0.8655	0.0822
PMF Sigmoid (k=50;λ=0.01;ϕ=0.01)	0.6499	0.7223	0.0104	0.7235	0.7223	0.0559
Maximum Cosine Similarity	0.7055	0.9166	0.0126	0.8706	0.9166	0.0703
PMF Basic (k=50;λ=0.01;ϕ=0.01)	0.7137	0.9039	0.0126	0.8316	0.9039	0.0718
Random	**0.7311**	0.8922	0.0152	0.8661	0.8922	0.0849

Table 41. Comparison of the different algorithms in terms of community Gini (1 Month follows graph) (2 of 2)

200 Tweets

Finally, we show the results for the second dataset.

Interactions

In this section, we show the complete results for the interactions graph of this dataset. We divide the results in 9 different tables.

First, in Table 42, we show the values of the accuracy metrics. Then, tables 43 and 44 show the values for the novelty and diversity metrics. Tables 45 and 46 display the values for embeddedness, clustering coefficient and distance metrics. Tables 47 and 48 show the modularity of the extended graphs, the number of weak ties and the degree assortativity. Finally, tables 49 and 50 show the community Gini coefficients.

Algorithm	P@10	R@10	nDCG@10	Algorithm	P@10	R@10	nDCG@10
BM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.3;k=1.0)	**0.02385**	0.05445	**0.04540**	Hub Promoted Index ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.01179	0.02604	0.02095
ExtremeBM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.1)	0.02367	0.05417	0.04526	TF-IDF ($\Gamma_{und}(u)$; $\Gamma_{und}(v)$)	0.01170	0.02605	0.02174
BIR ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.02330	0.05339	0.04422	Money (Auth;k=500;α=0.3)	0.01161	0.02831	0.02221
Adamic/Adar ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.02325	0.05354	0.04380	Popularity	0.00983	0.02206	0.01844
ImplicitMF (k=300;λ=150;α=40)	0.02260	**0.05736**	0.04450	Preferential Attachment ($\Gamma_{und}(v)$)	0.00983	0.02211	0.01845
FOAF ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.02211	0.05036	0.04192	PageRank (r=0.8)	0.00887	0.02066	0.01788
QLJM ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;λ=0.1)	0.02100	0.04709	0.03954	Love (Hubs;k=10;α=0.1)	0.00872	0.02038	0.01438
Res. Allocation ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.02090	0.04872	0.03995	Salton ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.00853	0.01988	0.01718
Personalized SALSA (Auth.;α=0.99)	0.02085	0.05166	0.04079	Personalized PageRank (r=0.4)	0.00682	0.01720	0.01492
User-Based kNN (k=100)	0.02021	0.05144	0.04096	Maximum Cosine Similarity	0.00602	0.01538	0.01264
Centroid Cosine Similarity	0.01791	0.04358	0.03573	Local Path Index ($\Gamma_{in}(u)$;β=0.1;l=3)	0.00569	0.01463	0.01190
Hannon ($\Gamma_{in}(u)$)	0.01690	0.03753	0.03169	Distance (Dir.)	0.00560	0.01349	0.00981
Average Cosine Similarity	0.01619	0.03874	0.03248	Commute Time	0.00544	0.01407	0.01127
Matrix Forest (α=0.001)	0.01540	0.04015	0.03067	LHN Index 1 ($\Gamma_{in}(u)$; $\Gamma_{out}(v)$)	0.00537	0.01196	0.00961
Item-Based kNN (k=290)	0.01508	0.03646	0.03092	Hitting Time	0.00515	0.01366	0.01087
Pure Personalized PageRank (r=0.4)	0.01476	0.03750	0.02791	LHN Index 2 (β=0.1)	0.00339	0.00723	0.00526
PropFlow ($\Gamma_{out}(u)$;l=5)	0.01403	0.03572	0.02671	Katz ($\Gamma_{out}(u)$;β=0.1)	0.00322	0.00971	0.00755
Hub Depressed Index ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.01387	0.03207	0.02668	PMF Sigmoid (k=20;λ=0.01;ϕ=0.01)	0.00056	0.00092	0.00055
Jaccard ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.01331	0.03095	0.02566	PMF Basic (k=60;λ=0.01;ϕ=0.01)	0.00023	0.00059	0.00042
Sorensen ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.01331	0.03095	0.02566	Random	0.00021	0.00033	0.00036
Personalized HITS (Auth.;α=0.99)	0.01195	0.02662	0.02196				

Table 42. Comparison of the different algorithms in terms of P@10. R@10 and nDCG@10 (200 Tweets interactions graph)

Algorithm	Novelty		Diversity		ERR-IA			Subtopic-Recall		
	PC	PD	ILD	Gini	Infomap	Louvain	Leading Vector	Infomap	Louvain	Leading Vector
BM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.3;k=1.0)	0.9838	0.5322	0.5400	0.1764	0.0218	0.0259	0.0279	**0.0010**	0.0167	**0.0417**
ExtremeBM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.1)	0.9835	0.5314	0.5363	0.1763	**0.0219**	**0.0260**	**0.0281**	0.0010	0.0166	0.0411
BIR ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.9827	0.5377	0.5514	0.1630	0.0212	0.0252	0.0272	0.0010	0.0163	0.0406
Adamic/Adar ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.9829	0.5417	0.5532	0.1806	0.0209	0.0251	0.0268	0.0010	0.0163	0.0405
ImplicitMF (k=300;λ=150;α=40)	0.9848	0.4880	0.5136	0.0595	0.0208	0.0245	0.0260	0.0010	0.0165	0.0410
FOAF ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.9835	0.5404	0.5392	0.1535	0.0202	0.0240	0.0257	0.0009	0.0154	0.0386
QLJM ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;λ=0.1)	0.9897	0.5318	0.5158	0.3040	0.0195	0.0236	0.0250	0.0009	0.0146	0.0363
Res. Allocation ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.9851	0.5746	0.6193	0.2289	0.0192	0.0233	0.0247	0.0009	0.0152	0.0375
Personalized SALSA (Auth.;α=0.99)	0.9711	0.5328	0.6177	0.0656	0.0187	0.0222	0.0235	0.0009	0.0157	0.0386
User-Based kNN (k=100)	0.9755	0.5176	0.5503	0.0528	0.0196	0.0231	0.0242	0.0009	0.0150	0.0375
Centroid Cosine Similarity	0.9835	0.4753	0.4240	0.1177	0.0186	0.0217	0.0219	0.0008	0.0129	0.0330
Hannon ($\Gamma_{in}(u)$)	0.9918	0.5531	0.4489	0.2590	0.0163	0.0195	0.0209	0.0007	0.0117	0.0297
Average Cosine Similarity	0.9902	0.4891	0.4646	0.1956	0.0176	0.0207	0.0207	0.0007	0.0116	0.0299
Matrix Forest (α=0.001)	0.9923	0.5305	0.5748	0.2392	0.0147	0.0177	0.0180	0.0007	0.0120	0.0302
Item-Based kNN (k=290)	0.9907	0.5068	0.5046	0.1886	0.0171	0.0200	0.0200	0.0007	0.0110	0.0283
Pure Personalized PageRank (r=0.4)	0.9786	0.6038	0.6995	0.0918	0.0119	0.0140	0.0153	0.0007	0.0120	0.0297
PropFlow ($\Gamma_{out}(u)$;l=5)	0.9821	0.6077	0.7192	0.0777	0.0114	0.0135	0.0145	0.0007	0.0114	0.0279
Hub Depressed Index ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.9977	0.5542	0.5339	0.4340	0.0131	0.0163	0.0168	0.0006	0.0101	0.0252
Jaccard ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.9981	0.5562	0.5184	0.4446	0.0125	0.0154	0.0160	0.0006	0.0095	0.0239
Sorensen ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.9981	0.5562	0.5184	0.4446	0.0125	0.0154	0.0160	0.0006	0.0095	0.0239
Personalized HITS (Auth.;α=0.99)	0.9455	0.7845	0.6761	0.0013	0.0051	0.0233	0.0111	0.0005	**0.0232**	0.0256

Table 43. Comparison of the different algorithms in terms novelty and diversity (200 Tweets interactions graph) (1 of 2)

	Novelty		Diversity		ERR-IA			Subtopic Recall		
Algorithm	PC	PD	ILD	Gini	Infomap	Louvain	Leading Vector	Infomap	Louvain	Leading Vector
Hub Promoted Index ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.9946	0.6112	0.5741	0.2859	0.0100	0.0123	0.0131	0.0006	0.0090	0.0226
TF-IDF ($\Gamma_{und}(u);\Gamma_{und}(v)$)	0.9988	0.5267	0.4788	0.5088	0.0110	0.0137	0.0144	0.0005	0.0084	0.0212
Money (Auth;k=500;α=0.3)	0.9771	0.5921	0.5795	0.0273	0.0103	0.0119	0.0127	0.0005	0.0094	0.0231
Popularity	0.9387	0.7941	0.6319	0.0010	0.0053	0.0067	0.0077	0.0005	0.0090	0.0227
Preferential Attachment ($\Gamma_{und}(v)$)	0.9387	0.7941	0.6319	0.0010	0.0053	0.0067	0.0077	0.0005	0.0090	0.0227
PageRank (r=0.8)	0.9433	0.7994	0.7191	0.0010	0.0053	0.0063	0.0078	0.0005	0.0082	0.0207
Love (Hubs;k=10;α=0.1)	0.9446	0.8018	0.7374	0.0010	0.0041	0.0118	0.0054	0.0005	0.0185	0.0205
Salton ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.9990	0.5945	0.5808	0.3130	0.0092	0.0114	0.0115	0.0004	0.0061	0.0155
Personalized PageRank (r=0.4)	0.9601	0.8056	0.7745	0.0009	0.0046	0.0052	0.0070	0.0004	0.0065	0.0162
Maximum Cosine Similarity	0.9974	0.6076	0.6215	0.2169	0.0061	0.0075	0.0072	0.0003	0.0049	0.0125
Local Path Index ($\Gamma_{in}(u)$;β=0.1;l=3)	0.9967	0.6015	0.4882	0.1473	0.0052	0.0067	0.0068	0.0003	0.0050	0.0127
Distance (Dir.)	0.9941	0.6331	0.7640	0.3213	0.0046	0.0055	0.0060	0.0003	0.0049	0.0121
Commute Time	0.9699	0.8433	0.9277	0.0010	0.0031	0.0039	0.0048	0.0003	0.0054	0.0130
LHN Index 1 ($\Gamma_{in}(u);\Gamma_{out}(v)$)	0.9989	0.6502	0.6513	0.2291	0.0045	0.0058	0.0063	0.0003	0.0045	0.0112
Hitting Time	0.9702	**0.8491**	0.9312	0.0009	0.0028	0.0036	0.0045	0.0003	0.0051	0.0125
LHN Index 2 (β=0.1)	**0.9998**	0.6428	0.5967	0.1324	0.0022	0.0031	0.0033	0.0002	0.0027	0.0070
Katz ($\Gamma_{out}(u)$;β=0.1)	0.9862	0.8182	0.5565	0.0099	0.0029	0.0035	0.0034	0.0002	0.0029	0.0071
PMF Sigmoid (k=20;λ=0.01;ϕ=0.01)	0.9957	0.8235	0.9343	0.0009	0.0002	0.0002	0.0002	0.0000	0.0006	0.0014
PMF Basic (k=60;λ=0.01;ϕ=0.01)	0.9990	0.8298	0.9332	0.0097	0.0002	0.0002	0.0002	0.0000	0.0002	0.0006
Random	0.9985	0.8401	**0.9512**	**0.8278**	0.0002	0.0003	0.0003	0.0000	0.0002	0.0005

Table 44. Comparison of the different algorithms in terms novelty and diversity (200 Tweets interactions graph) (2 of 2)

	Clustering Coefficient		Embeddedness		Distances				
Algorithm	Global	Average	Global	Zero	ASL	Avg. Ecc.	Diameter	Rec. Dist	Edges ∞
BM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.3;k=1.0)	0.2135	0.3229	0.0471	34,332	4.2038	6.6704	9	2.4023	12,648
ExtremeBM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.1)	0.2107	0.3240	0.0471	34,214	4.1677	6.6453	9	2.3968	12,648
BIR ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.1937	0.3301	0.0439	35,087	4.0103	6.4065	8	2.3828	12,651
Adamic/Adar ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.1941	0.3490	0.0459	34,875	3.9829	6.3643	8	2.4675	12,661
ImplicitMF (k=300;λ=150;α=40)	0.1650	0.2355	0.0335	41,170	3.8311	6.0754	7	3.0264	12,694
FOAF ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.1944	0.3568	0.0454	37,402	4.0066	6.4389	8	2.5551	12,687
QLJM ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;λ=0.1)	0.1919	0.3021	0.0632	31,941	3.9769	6.2152	8	2.6373	12,690
Res. Allocation ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.1786	0.3372	0.0455	32,581	3.8874	6.1710	8	2.5249	12,661
Personalized SALSA (Auth.;α=0.99)	0.1847	0.2794	0.0319	55,953	3.8700	6.3174	8	2.5881	12,679
User-Based kNN (k=100)	0.1685	0.2513	0.0309	54,837	3.8868	6.3655	8	2.6749	12,684
Centroid Cosine Similarity	0.1694	0.1914	0.0393	58,906	3.8846	6.0066	7	2.7934	12,689
Hannon ($\Gamma_{in}(u)$)	0.2034	0.3456	0.0757	35,586	4.1840	6.7382	10	2.8471	12,706
Average Cosine Similarity	0.1697	0.1891	0.0493	48298	3.9123	6.0396	8	2.9208	12,694
Matrix Forest (α=0.001)	0.1647	0.2130	0.0499	32,699	4.0609	6.6671	9	2.0482	12,634
Item-Based kNN (k=290)	0.1567	0.1897	0.0461	55,993	3.9789	6.5187	8	2.6436	12,686
Pure Personalized PageRank (r=0.4)	0.1721	0.2701	0.0329	36,471	4.1008	6.6797	9	2.0991	12,634
PropFlow ($\Gamma_{out}(u)$;l=5)	0.1613	0.2441	0.0143	48,794	4.0095	6.5133	8	2.1416	12,634
Hub Depressed Index ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.1750	0.2737	0.0720	31,770	3.9779	6.1818	8	2.8423	12,697
Jaccard ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.1901	0.2929	0.0866	30,920	3.9638	5.9364	8	2.9692	12,697
Sorensen ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.1901	0.2929	0.0866	30,920	3.9638	5.9364	8	2.9692	12,697
Personalized HITS (Auth.;α=0.99)	0.2274	0.4126	0.0165	57,135	3.9419	6.5539	7	2.9345	12,710

Table 45. Comparison of the different algorithms in terms of clustering coefficient. embeddedness and distance (200 Tweets interactions graph) (1 of 2)

	Clustering Coefficient		Embeddedness		Distances				
Algorithm	Global	Average	Global	Zero	ASL	Avg. Ecc.	Diameter	Rec. Dist	Edges ∞
Hub Promoted Index ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.1398	0.2623	0.0504	51,146	3.8527	6.0668	8	3.3231	12,983
TF-IDF ($\Gamma_{und}(u);\Gamma_{und}(v)$)	0.1766	0.2269	0.0738	38,015	3.9469	5.8876	9	3.4200	13,663
Money (Auth;k=500;α=0.3)	0.1724	0.2289	0.0237	61,618	3.9552	6.4180	8	2.6971	12,708
Popularity	0.2995	0.6291	0.0139	52,001	3.9679	6.7155	7	3.0832	12,710
Preferential Attachment ($\Gamma_{und}(v)$)	0.2995	0.6291	0.0143	48,794	3.9679	6.7155	7	3.0832	12,710
PageRank (r=0.8)	0.2983	0.6287	0.0141	49,620	3.8745	5.9796	7	3.1875	12,710
Love (Hubs;k=10;α=0.1)	0.2916	0.6276	0.0142	49,004	3.8459	5.9796	7	3.3031	12,710
Salton ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.1899	0.2922	0.0829	35,373	3.9107	5.7923	8	3.2010	12,697
Personalized PageRank (r=0.4)	0.3973	0.6250	0.0140	50,793	4.2050	6.7007	7	3.0804	12,710
Maximum Cosine Similarity	0.1092	0.1453	0.0374	58,036	3.7867	5.7144	7	3.4535	12,697
Local Path Index ($\Gamma_{in}(u)$;β=0.1;l=3)	0.1337	0.2958	0.0561	44,451	3.4705	6.0337	7	3.2857	12,801
Distance (Dir.)	0.0908	0.1236	0.0276	29,751	3.8946	6.4310	8	2.0482	12,634
Commute Time	0.4276	0.6163	0.0141	51,755	4.5067	7.0665	9	3.1353	12,676
LHN Index 1 ($\Gamma_{in}(u);\Gamma_{out}(v)$)	0.1555	0.2458	0.0538	57,412	3.8145	6.1190	8	3.4972	13,225
Hitting Time	0.4379	0.6214	0.0139	52,046	4.5113	7.0675	9	3.1515	12,671
LHN Index 2 (β=0.1)	0.1847	0.2507	0.0521	51,596	4.2094	6.3664	10	3.8137	**14,519**
Katz ($\Gamma_{out}(u)$;β=0.1)	0.1517	0.2944	0.0183	50,299	3.8743	6.2588	8	3.3607	12,634
PMF Sigmoid (k=20;λ=0.01;ϕ=0.01)	0.4252	0.6239	0.0138	71,648	4.2956	6.7880	8	4.3857	12,710
PMF Basic (k=60;λ=0.01;ϕ=0.01)	0.1107	0.1542	0.0620	39,964	3.8035	6.1358	7	**4.8903**	12,710
Random	**0.0431**	**0.0296**	**0.0118**	**144,084**	**3.3479**	**4.6382**	**5**	4.7038	13,275

Table 46. Comparison of the different algorithms in terms of clustering coefficient, embeddedness and distance (200 Tweets interactions graph) (2 of 2)

Algorithm	Assortativity		Modularity			Weak ties		
	Undirected	Directed (IN)	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
BM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.3;k=1.0)	-0.0569	0.0331	0.5864	0.5000	0.4902	33,353	34,419	48,042
ExtremeBM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.1)	-0.0604	0.0308	0.5865	0.5002	0.4898	33,317	34,443	48,122
BIR ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	-0.0747	0.0200	0.5791	0.4939	0.4798	34,720	35,648	50,271
Adamic/Adar ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	-0.0694	0.0215	0.5820	0.4907	0.4838	34,175	36,122	49,481
ImplicitMF (k=300;λ=150;α=40)	-0.1762	-0.0380	0.5647	0.4838	0.4632	37,767	37,572	53,802
FOAF ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	-0.0785	0.0211	0.5823	0.4962	0.4840	34,112	35,231	49,376
QLJM ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;λ=0.1)	-0.0565	0.0414	0.5975	0.5008	0.5037	31,152	34,385	45,394
Res. Allocation ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	-0.0525	0.0296	0.5574	0.4726	0.4624	39,194	38,827	54,371
Personalized SALSA (Auth.;α=0.99)	-0.1056	-0.0269	0.5279	0.4508	0.4358	44,914	42,731	59,603
User-Based kNN (k=100)	-0.1082	-0.0235	0.5413	0.4699	0.4468	42,212	39,732	57,132
Centroid Cosine Similarity	-0.1515	-0.0365	0.5593	0.4767	0.4690	38,566	38,940	52,513
Hannon ($\Gamma_{in}(u)$)	-0.0834	0.0345	0.4457	0.3885	0.3657	61,446	53,680	48,777
Average Cosine Similarity	-0.1296	-0.0327	0.5739	0.4756	0.4890	35,996	38,889	48,774
Matrix Forest (α=0.001)	-0.1342	-0.0321	0.5509	0.4698	0.4574	40,858	38,849	55,881
Item-Based kNN (k=290)	-0.0993	0.0017	0.5635	0.4669	0.4793	37,984	39,798	50,788
Pure Personalized PageRank (r=0.4)	-0.0923	-0.0186	0.4859	0.4384	0.3968	53,562	44,194	68,688
PropFlow ($\Gamma_{out}(u)$;l=5)	-0.0959	-0.0177	0.4855	0.4372	0.3941	53,707	43,944	69,364
Hub Depressed Index ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	-0.0050	0.0416	0.5921	0.4928	0.4991	32,359	35,397	46,873
Jaccard ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	-0.0010	0.0418	0.5877	0.4884	0.4993	33,228	36,177	46,782
Sorensen ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	-0.0010	0.0418	0.5877	0.4884	0.4993	33,228	36,177	46,782
Personalized HITS (Auth.;α=0.99)	-0.2240	-0.0318	0.3900	0.3383	0.3211	71,666	63,741	83,115

Table 47. Comparison of the different algorithms in terms of assortativity, modularity and weak ties (200 Tweets interactions graph) (1 of 2)

Algorithm	Assortativity		Modularity			Weak ties		
	Undirected	Directed (IN)	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
Hub Promoted Index ($\Gamma_{und}(u);\Gamma_{in}(v)$)	-0.0654	-0.0337	0.5585	0.4659	0.4603	39,271	39,584	55,803
TF-IDF ($\Gamma_{und}(u);\Gamma_{und}(v)$)	-0.0037	0.0199	0.6150	0.5046	0.5288	27,997	33,432	40,737
Money (Auth;k=500;α=0.3)	-0.1534	-0.0260	0.4853	0.4367	0.3923	53,424	44,901	68,998
Popularity	**-0.2821**	-0.0275	**0.3259**	**0.2613**	**0.2772**	84,208	78,279	92,416
Preferential Attachment ($\Gamma_{und}(v)$)	**-0.2821**	-0.0275	**0.3259**	**0.2613**	**0.2772**	84,208	78,279	92,416
PageRank (r=0.8)	-0.2812	-0.0272	0.3266	0.2674	0.2787	85,259	75,207	92,637
Love (Hubs;k=10;α=0.1)	-0.2807	-0.0285	0.3268	0.2701	0.2785	85,205	75,012	92,826
Salton ($\Gamma_{und}(u);\Gamma_{in}(v)$)	-0.0178	0.0233	0.5519	0.4679	0.4664	40,381	39,557	54,037
Personalized PageRank (r=0.4)	-0.2785	-0.0123	0.3308	0.2853	0.2808	84,965	69,091	93,035
Maximum Cosine Similarity	-0.1023	-0.0232	0.5022	0.4320	0.4081	50,598	45,814	67,294
Local Path Index ($\Gamma_{in}(u)$;β=0.1;l=3)	-0.0633	-0.0215	0.5609	0.4620	0.4626	38,682	40,636	55,191
Distance (Dir.)	-0.1224	-0.0385	0.4751	0.4304	0.3792	55,996	45,814	72,828
Commute Time	-0.2699	-0.0062	0.3424	0.3012	0.2832	84,172	63,234	95,753
LHN Index 1 ($\Gamma_{in}(u);\Gamma_{out}(v)$)	-0.0802	**-0.0507**	0.5339	0.4518	0.4345	44,246	41,906	61,964
Hitting Time	-0.2743	-0.0052	0.3379	0.2987	0.2810	85,168	63,633	96,341
LHN Index 2 (β=0.1)	-0.0817	-0.0269	0.5495	0.4874	0.4479	41,579	35,902	59,536
Katz ($\Gamma_{out}(u)$;β=0.1)	-0.1982	-0.0215	0.3532	0.2955	0.2954	80,129	67,612	93,009
PMF Sigmoid (k=20;λ=0.01;ϕ=0.01)	-0.2738	-0.0076	0.3353	0.2853	0.2833	83,643	71,036	93,962
PMF Basic (k=60;λ=0.01;ϕ=0.01)	-0.1589	-0.0121	0.3368	0.2935	0.2827	83,949	68,951	94,345
Random	-0.0222	-0.0196	0.3359	0.2950	0.2818	83,949	67,456	95,154

Table 48. Comparison of the different algorithms in terms of assortativity, modularity and weak ties (200 Tweets interactions graph) (2 of 2)

Algorithm	Comm-PairGini			Comm-InGini		
	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
BM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.3;k=1.0)	0.3229	0.6966	0.0341	0.5290	0.8712	0.1462
ExtremeBM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.1)	0.3212	**0.6968**	0.0338	0.5262	0.8779	0.1444
BIR ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.3190	0.6929	0.0330	0.5162	**0.8907**	0.1382
Adamic/Adar ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.3197	0.6955	0.0326	0.5229	0.8806	0.1431
ImplicitMF (k=300;λ=150;α=40)	0.3522	0.6847	0.0342	0.5629	0.8443	0.1245
FOAF ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.3142	0.6908	0.0324	0.5149	0.8853	0.1404
QLJM ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;λ=0.1)	0.3216	0.6961	0.0355	0.5359	0.8765	0.1642
Res. Allocation ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.3291	0.6947	0.0361	0.5370	0.8457	0.1634
Personalized SALSA (Auth.;α=0.99)	0.3288	0.6741	0.0335	0.5080	0.8723	0.1276
User-Based kNN (k=100)	0.3331	0.6703	0.0337	0.5114	0.8729	0.1185
Centroid Cosine Similarity	0.3215	0.6590	0.0352	0.4926	0.8490	0.1350
Hannon ($\Gamma_{in}(u)$)	0.3025	0.6332	0.0328	0.4337	0.7270	0.1612
Average Cosine Similarity	0.3468	0.6795	0.0417	0.5601	0.8476	0.1761
Matrix Forest (α=0.001)	0.3509	0.6648	0.0416	0.6029	0.8098	0.1620
Item-Based kNN (k=290)	0.3405	0.6701	0.0405	0.5462	0.8517	0.1725
Pure Personalized PageRank (r=0.4)	0.3377	0.6680	0.0352	0.5224	0.8332	0.1385
PropFlow ($\Gamma_{out}(u)$;l=5)	0.3515	0.6608	0.0391	0.5620	0.7969	0.1440
Hub Depressed Index ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.3292	0.6900	0.0401	0.5589	0.8412	0.1996
Jaccard ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.3288	0.6906	0.0386	0.5480	0.8401	0.2037
Sorensen ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.3288	0.6906	0.0386	0.5480	0.8401	0.2037
Personalized HITS (Auth.;α=0.99)	0.2865	0.6264	0.0763	0.3864	0.7397	0.0219

Table 49. Comparison of the different algorithms in terms of community Gini (200 Tweets interactions graph) (1 of 2)

Algorithm	Comm-PairGini			Comm-InGini		
	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
Hub Promoted Index ($\Gamma_{und}(u); \Gamma_{in}(v)$)	0.3371	0.6792	0.0417	0.5735	0.7872	0.2254
TF-IDF ($\Gamma_{und}(u); \Gamma_{und}(v)$)	0.3359	0.6875	0.0460	0.5924	0.8044	0.2465
Money (Auth;k=500;α=0.3)	0.3223	0.6469	0.0352	0.4948	0.8188	0.1174
Popularity	0.2574	0.6282	0.0165	0.3282	0.7116	0.0651
Preferential Attachment ($\Gamma_{und}(v)$)	0.2574	0.6282	0.0165	0.3282	0.7116	0.0651
PageRank (r=0.8)	0.3021	0.6477	0.0186	0.4000	0.7549	0.0699
Love (Hubs;k=10;α=0.1)	0.3092	0.6504	0.0695	0.4143	0.7681	0.0184
Salton ($\Gamma_{und}(u); \Gamma_{in}(v)$)	0.3267	0.6827	0.0391	0.5297	0.8296	0.2134
Personalized PageRank (r=0.4)	0.2813	0.5642	0.0197	0.3729	0.6673	0.0724
Maximum Cosine Similarity	0.3559	0.6737	0.0462	0.5755	0.8129	0.2167
Local Path Index ($\Gamma_{in}(u)$;β=0.1;l=3)	0.3286	0.6918	0.0361	0.5337	0.8397	0.1713
Distance (Dir.)	0.3540	0.6891	0.0417	0.5764	0.8551	0.1558
Commute Time	0.2893	0.5334	0.0261	0.3903	0.5757	0.0901
LHN Index 1 ($\Gamma_{in}(u); \Gamma_{out}(v)$)	0.3362	0.6698	0.0477	0.5581	0.7647	0.2539
Hitting Time	0.2869	0.5328	0.0260	0.3857	0.5750	0.0900
LHN Index 2 (β=0.1)	0.3491	0.6651	0.0512	**0.6211**	0.7335	**0.2761**
Katz ($\Gamma_{out}(u)$;β=0.1)	0.2968	0.6102	0.0309	0.4114	0.7117	0.1037
PMF Sigmoid (k=20;λ=0.01;ϕ=0.01)	0.3291	0.6611	0.0253	0.4613	0.8081	0.0882
PMF Basic (k=60;λ=0.01;ϕ=0.01)	0.3817	0.6677	0.0477	0.5772	0.8239	0.1390
Random	**0.3873**	0.6396	**0.0797**	0.5757	0.7631	0.2354

Table 50. Comparison of the different algorithms in terms of community Gini (200 Tweets interactions graph) (2 of 2)

Follows

In this section, we show the complete results for the explicit follows graph of this dataset. We divide the results in 9 different tables.

First, in Table 51Table 42, we show the values of the accuracy metrics. Then, tables 52 and 53 show the values for the novelty and diversity metrics. Tables 54 and 55 display the values for embeddedness, clustering coefficient and distance metrics. Tables 56 and 57 show the modularity of the extended graphs, the number of weak ties and the degree assortativity. Finally, tables 58 and 59 show the community Gini coefficients.

Algorithm	P@10	R@10	nDCG@10	Algorithm	P@10	R@10	nDCG@10
ImplicitMF (k=300;α=40;λ=150)	**0.03648**	**0.06916**	**0.06005**	Maximum Cosine Similarity	0.02286	0.04311	0.03746
User-based kNN (k=40)	0.03428	0.06187	0.05624	Salton ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.02283	0.04071	0.03633
Average Cosine Similarity	0.03385	0.06145	0.05492	Hub Depressed Index ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.02213	0.03892	0.03552
Adamic ($\Gamma_{und}(u);\Gamma_{in}(v);\Gamma_{und}(w)$)	0.03283	0.05680	0.05180	Hub Promoted Index ($\Gamma_{in}(u);\Gamma_{und}(v)$)	0.01916	0.03594	0.03085
QLJM ($\Gamma_{und}(u);\Gamma_{in}(v)$;$\lambda$=0.1)	0.03275	0.05730	0.05209	Money (Auth; k=50; α=0.3)	0.01293	0.02029	0.01945
BM25 ($\Gamma_{und}(u);\Gamma_{in}(v)$;$b$=0.1;$k$=1)	0.03263	0.05684	0.05157	Preferential Attachment ($\Gamma_{und}(v)$)	0.01175	0.01922	0.01813
BIR ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.03226	0.05624	0.05115	Popularity	0.01170	0.01893	0.01745
Item-based kNN (k=300)	0.03218	0.05797	0.05301	PageRank (r=0.8)	0.00915	0.01569	0.01375
Res. Allocation ($\Gamma_{und}(u);\Gamma_{in}(v);\Gamma_{und}(w)$)	0.03217	0.05761	0.05165	Personalized PageRank (r=0.8)	0.00915	0.01569	0.01375
ExtremeBM25 ($\Gamma_{und}(u);\Gamma_{in}(v)$;$b$=0.1)	0.03214	0.05712	0.05087	LHN Index 1 ($\Gamma_{in}(u);\Gamma_{out}(v)$)	0.00622	0.01296	0.00969
Centroid Cosine Similarity	0.03162	0.05571	0.05084	Distance (Dir.)	0.00321	0.00658	0.00483
FOAF ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.03161	0.05418	0.04966	Katz ($\Gamma_{out}(u)$;β=0.4)	0.00295	0.00593	0.00507
Matrix Forest (α=0.001)	0.03056	0.05503	0.04890	LHN Index 2 (β=0.2)	0.00282	0.00757	0.00477
Personalized SALSA (Auth;α=0.99)	0.03039	0.05673	0.04885	Hitting Time	0.00273	0.00572	0.00433
Local Path Index ($\Gamma_{out}(u)$;l=2;β=0.2)	0.02957	0.05141	0.04677	Commute Time	0.00270	0.00558	0.00454
Pure Personalized PageRank (r=0.8)	0.02449	0.04543	0.03766	Personalized HITS (Auth.;α=0.99)	0.00158	0.00181	0.00217
Jaccard ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.02394	0.04290	0.03848	Love(Auth.;k=1000;α=0.2)	0.00147	0.00164	0.00205
Sorensen ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.02394	0.04290	0.03848	PMF Basic (k=20.0;λ=0.01;ϕ=0.01)	0.00071	0.00099	0.00102
PropFlow ($\Gamma_{out}(u)$;l=2)	0.02388	0.04558	0.03720	Random	0.00048	0.00085	0.00063
TF-IDF ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.02329	0.04234	0.03759	PMF Sigmoid (k=20.0;λ=0.01;ϕ=0.01)	0.00044	0.00074	0.00062
Hannon ($\Gamma_{in}(u)$)	0.02312	0.04231	0.03760				

Table 51. Comparison of the different algorithms in terms of P@10, R@10 and nDCG@10 (200 Tweets follows graph)

Algorithm	Novelty		Diversity		ERR-IA			Subtopic-Recall		
	PC	PD	ILD	Gini	Infomap	Louvain	Leading Vector	Infomap	Louvain	Leading Vector
ImplicitMF (k=300;α=40;λ=150)	0.9694	0.4014	0.3746	0.1174	**0.0369**	**0.0371**	**0.0411**	0.0036	**0.0381**	**0.0883**
User-based kNN (k=40)	0.9599	0.3896	0.3053	0.0860	0.0367	0.0363	0.0399	0.0032	0.0336	0.0784
Average Cosine Similarity	0.9624	0.3469	0.2100	0.1126	0.0361	0.0359	0.0389	0.0031	0.0326	0.0771
Adamic ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.9516	0.4030	0.3253	0.0948	0.0330	0.0326	0.0360	0.0031	0.0331	0.0769
QLJM ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;λ=0.1)	0.9662	0.3904	0.2962	0.1705	0.0333	0.0334	0.0363	0.0031	0.0328	0.0772
BM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.1;k=1)	0.9524	0.4000	0.3223	0.0960	0.0326	0.0323	0.0357	0.0031	0.0330	0.0772
BIR ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.9511	0.4005	0.3193	0.0872	0.0329	0.0323	0.0358	0.0031	0.0326	0.0756
Item-based kNN (k=300)	0.9621	0.3743	0.2700	0.1044	0.0355	0.0352	0.0383	0.0030	0.0316	0.0749
Res. Allocation ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.9616	0.4409	0.3997	0.1693	0.0317	0.0318	0.0351	0.0032	0.0333	0.0772
ExtremeBM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.1)	0.9538	0.4024	0.3304	0.0980	0.0316	0.0314	0.0347	0.0031	0.0326	0.0764
Centroid Cosine Similarity	0.9525	0.3590	0.2024	0.0685	0.0331	0.0325	0.0360	0.0029	0.0306	0.0723
FOAF ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.9495	0.4031	0.3155	0.0804	0.0321	0.0316	0.0350	0.0030	0.0319	0.0741
Matrix Forest (α=0.001)	0.9761	0.3825	0.3064	0.1681	0.0318	0.0316	0.0345	0.0030	0.0312	0.0732
Personalized SALSA (Auth;α=0.99)	0.9332	0.4106	0.3959	0.0380	0.0303	0.0295	0.0337	0.0031	0.0324	0.0752
Local Path Index ($\Gamma_{out}(u)$;l=2;β=0.2)	0.9555	0.4018	0.3370	0.0733	0.0309	0.0302	0.0334	0.0029	0.0307	0.0713
Pure Personalized PageRank (r=0.8)	0.9642	0.4786	0.4989	0.0953	0.0219	0.0216	0.0244	0.0025	0.0267	0.0616
Jaccard ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.9918	0.4170	0.3351	0.4692	0.0251	0.0254	0.0275	0.0024	0.0250	0.0593
Sorensen ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.9918	0.4170	0.3351	0.4692	0.0251	0.0254	0.0275	0.0024	0.0250	0.0593
PropFlow ($\Gamma_{out}(u)$;l=2)	0.9713	0.4830	0.5121	0.1116	0.0214	0.0213	0.0238	0.0025	0.0260	0.0604
TF-IDF ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.9925	0.4423	0.3950	**0.5251**	0.0250	0.0253	0.0272	0.0023	0.0243	0.0580
Hannon ($\Gamma_{in}(u)$)	0.9805	0.4478	0.3257	0.3034	0.0238	0.0239	0.0261	0.0023	0.0244	0.0574

Table 52. Comparison of the different algorithms in terms novelty and diversity (200 Tweets follows graph) (1 of 2)

	Novelty		Diversity		ERR-IA			Subtopic Recall		
Algorithm	PC	PD	ILD	Gini	Infomap	Louvain	Leading Vector	Infomap	Louvain	Leading Vector
Maximum Cosine Similarity	0.9713	0.4304	0.3646	0.1118	0.0229	0.0228	0.0256	0.0023	0.0248	0.0578
Salton ($\Gamma_{und}(u); \Gamma_{in}(v)$)	0.9918	0.4330	0.3771	0.4484	0.0246	0.0246	0.0265	0.0022	0.0236	0.0564
Hub Depressed Index ($\Gamma_{und}(u); \Gamma_{in}(v)$)	0.9920	0.4200	0.3424	0.4430	0.0234	0.0237	0.0257	0.0023	0.0235	0.0560
Hub Promoted Index ($\Gamma_{in}(u); \Gamma_{und}(v)$)	0.9763	0.4938	0.4484	0.2484	0.0194	0.0197	0.0210	0.0020	0.0211	0.0497
Money (Auth; k=50; α=0.3)	0.9362	0.5289	0.3483	0.0078	0.0126	0.0109	0.0135	0.0014	0.0151	0.0349
Preferential Attachment ($\Gamma_{und}(v)$)	0.8773	0.7328	0.5840	0.0012	0.0076	0.0061	0.0090	0.0014	0.0146	0.0338
Popularity	0.8772	0.7340	0.5831	0.0012	0.0073	0.0058	0.0086	0.0014	0.0146	0.0337
PageRank (r=0.8)	0.8984	0.7869	**0.7836**	0.0011	0.0053	0.0044	0.0068	0.0011	0.0121	0.0278
Personalized PageRank (r=0.8)	0.8984	0.7869	0.7836	0.0011	0.0053	0.0044	0.0068	0.0011	0.0121	0.0278
LHN Index 1 ($\Gamma_{in}(u); \Gamma_{out}(v)$)	0.9969	0.5908	0.6206	0.2009	0.0051	0.0055	0.0056	0.0007	0.0078	0.0185
Distance (Dir.)	0.9851	0.6000	0.6837	0.3849	0.0025	0.0024	0.0030	0.0004	0.0045	0.0105
Katz ($\Gamma_{out}(u)$;β=0.4)	0.9730	0.8497	0.7290	0.0067	0.0015	0.0016	0.0025	0.0004	0.0040	0.0093
LHN Index 2 (β=0.2)	**0.9996**	0.5001	0.5516	0.1227	0.0023	0.0026	0.0025	0.0003	0.0035	0.0084
Hitting Time	0.9640	0.8585	0.4010	0.0010	0.0011	0.0012	0.0020	0.0003	0.0033	0.0079
Commute Time	0.9643	**0.8607**	0.5216	0.0010	0.0012	0.0013	0.0023	0.0003	0.0033	0.0078
Personalized HITS (Auth.;α=0.99)	0.9175	0.0283	0.0004	0.0015	0.0020	0.0016	0.0021	0.0001	0.0015	0.0035
Love(Auth.;k=1000;α=0.2)	0.9200	0.0280	0.0001	0.0015	0.0020	0.0016	0.0021	0.0001	0.0015	0.0034
PMF Basic (k=20.0;λ=0.01;ϕ=0.01)	0.9961	0.7867	0.8328	0.0118	0.0006	0.0004	0.0007	0.0001	0.0010	0.0024
Random	0.9953	0.8047	0.8581	**0.8793**	0.0003	0.0003	0.0004	0.0001	0.0007	0.0016
PMF Sigmoid (k=20.0;λ=0.01;ϕ=0.01)	0.9934	0.8100	**0.8661**	0.0010	0.0002	0.0002	0.0003	0.0001	0.0006	0.0015

Table 53. Comparison of the different algorithms in terms novelty and diversity (200 Tweets follows graph) (2 of 2)

	Clustering Coefficient		Embeddedness		Distances				
Algorithm	Global	Average	Global	Zero	ASL	Avg. Ecc.	Diameter	Rec. Dist	Edges ∞
ImplicitMF (k=300;α=40;λ=150)	0.2053	0.2505	0.0628	22,006	3.2160	6.7032	8	2.1409	5,039
User-based kNN (k=40)	0.2118	0.2632	0.0623	25,830	3.1885	6.6790	8	2.1363	5,043
Average Cosine Similarity	0.2190	0.2453	0.0670	24,659	3.2226	6.3722	8	2.0919	5,034
Adamic ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.2219	0.3296	0.0620	21,028	3.1913	6.6766	8	2.0620	5,040
QLJM ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;λ=0.1)	0.2234	0.2997	0.0673	20,475	3.2073	6.6732	8	2.0666	5,037
BM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.1;k=1)	0.2308	0.3194	0.0612	22,252	3.2389	6.7027	8	2.0543	5,038
BIR ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.2223	0.3226	0.0613	21,265	3.1956	6.6683	8	2.0519	5,038
Item-based kNN (k=300)	0.2123	0.2427	0.0641	27,375	3.1838	6.3580	8	2.1599	5,045
Res. Allocation ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.2187	0.3315	0.0650	18,567	3.1788	6.6424	8	2.0746	5,039
ExtremeBM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.1)	0.2346	0.3180	0.0609	22,558	3.2582	6.7327	8	2.0536	5,038
Centroid Cosine Similarity	0.2165	0.2487	0.0608	27,858	3.2414	7.5080	10	2.2067	5,720
FOAF ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.2211	0.3235	0.0604	21,812	3.1901	6.6528	8	2.0648	5,039
Matrix Forest (α=0.001)	0.2144	0.2555	0.0687	19,332	3.2505	6.5935	8	2.0060	5,029
Personalized SALSA (Auth;α=0.99)	0.2212	0.3067	0.0570	26,518	3.1826	6.6343	8	2.0768	5,034
Local Path Index ($\Gamma_{out}(u)$;l=2;β=0.2)	0.2143	0.2739	0.0611	20,802	3.2309	6.5906	8	2.0111	5,139
Pure Personalized PageRank (r=0.8)	0.2050	0.2667	0.0588	18,804	3.2138	6.4341	8	2.0072	5,029
Jaccard ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.2174	0.2272	0.0790	18,262	3.2229	6.5777	9	2.1339	5,045
Sorensen ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.2174	0.2272	0.0790	18,262	3.2229	6.5777	9	2.1339	5,045
PropFlow ($\Gamma_{out}(u)$;l=2)	0.2021	0.2493	0.0594	19,213	3.1858	6.4908	8	2.0195	5,041
TF-IDF ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.2148	0.2280	0.0800	17,724	3.1636	5.9813	8	2.1897	5,055
Hannon ($\Gamma_{in}(u)$)	0.2084	0.2837	0.0711	20,165	3.2414	7.5080	10	2.2067	5,720

Table 54. Comparison of the different algorithms in terms of clustering coefficient, embeddedness and distance (200 Tweets follows graph) (1 of 2)

	Clustering Coefficient		Embeddedness		Distances				
Algorithm	Global	Average	Global	Zero	ASL	Avg. Ecc.	Diameter	Rec. Dist	Edges ∞
Maximum Cosine Similarity	0.2084	0.2837	0.0570	26,832	3.1736	5.9383	7	2.1503	5,038
Salton ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.1926	0.2036	0.0783	18,706	3.1915	6.4005	8	2.1844	5,052
Hub Depressed Index ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.2163	0.2251	0.0762	19,259	3.2248	6.4309	8	2.1275	5,044
Hub Promoted Index ($\Gamma_{in}(u);\Gamma_{und}(v)$)	0.2137	0.2252	0.0631	24,493	3.0996	6.0085	8	2.3592	8,455
Money (Auth; k=50; α=0.3)	0.1843	0.2857	0.0476	27,948	3.1601	6.6075	8	2.2176	5,040
Preferential Attachment ($\Gamma_{und}(v)$)	0.2032	0.2691	0.0449	25,761	3.1123	6.3792	7	2.3417	5,040
Popularity	0.2297	0.4518	0.0448	25,967	3.1193	6.4928	7	2.3435	5,040
PageRank (r=0.8)	0.2332	0.4519	0.0443	26,366	3.1467	6.6316	7	2.3338	5,040
Personalized PageRank (r=0.8)	0.2511	0.4513	0.0443	26,366	3.1467	6.6316	7	2.3339	5,040
LHN Index 1 ($\Gamma_{in}(u);\Gamma_{out}(v)$)	0.2511	0.4513	0.0597	31,775	3.1693	5.8220	7	2.6365	13,254
Distance (Dir.)	0.1767	0.2097	0.0456	17,773	3.1036	6.2343	8	2.0060	5,029
Katz ($\Gamma_{out}(u)$;β=0.4)	0.1513	0.1519	0.0438	29,907	3.3654	6.8187	9	2.7148	5,029
LHN Index 2 (β=0.2)	0.1890	0.2257	0.0571	29,794	3.2397	6.0459	8	2.8077	**45,762**
Hitting Time	0.1809	0.2082	0.0440	28,088	3.3136	6.7168	8	2.4545	5,029
Commute Time	0.2790	0.4476	0.0440	28,065	3.3096	6.7168	8	2.4772	5,029
Personalized HITS (Auth.;α=0.99)	0.2780	0.4479	0.0563	28,778	3.3357	6.5376	11	2.1867	133
Love(Auth.;k=1000;α=0.2)	0.1834	0.2041	0.0563	28,814	3.3344	6.5264	10	2.1790	161
PMF Basic (k=20.0;λ=0.01;ϕ=0.01)	0.1828	0.2037	0.0438	42,354	3.1077	6.6488	8	3.3532	5,041
Random	0.1651	0.1794	**0.0417**	**103,983**	**2.8904**	**4.2591**	**5**	3.3700	11,767
PMF Sigmoid (k=20.0;λ=0.01;ϕ=0.01)	**0.1443**	**0.0940**	0.0443	26,830	3.1648	6.7284	7	**3.5784**	5,040

Table 55. Comparison of the different algorithms in terms of clustering coefficient, embeddedness and distance (200 Tweets follows graph) (2 of 2)

Algorithm	Assortativity		Modularity			Weak ties		
	Undirected	Directed (IN)	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
ImplicitMF (k=300;α=40;λ=150)	-0.1482	-0.0405	0.5766	0.5924	0.5666	29,900	26,752	34,520
User-based kNN (k=40)	-0.1558	-0.0422	0.5806	0.5948	0.5735	28,005	25,871	31,425
Average Cosine Similarity	-0.1443	-0.0429	0.5881	0.6048	0.5850	24,627	22,399	26,593
Adamic ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	-0.1312	-0.0181	0.5779	0.5972	0.5735	28,849	24,925	31,052
QLJM ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;λ=0.1)	-0.1059	0.0025	0.5886	0.6015	0.5839	24,536	23,546	27,202
BM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.1;k=1)	-0.1296	-0.0126	0.5808	0.5982	0.5740	27,710	24,551	30,944
BIR ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	-0.1337	-0.0187	0.5781	0.5980	0.5722	28,806	24,600	31,508
Item-based kNN (k=300)	-0.1449	-0.0323	0.5798	0.5984	0.5768	28,198	24,626	30,015
Res. Allocation ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	-0.1100	-0.0067	0.5722	0.5864	0.5673	31,612	28,780	34,383
ExtremeBM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.1)	-0.1263	-0.0108	0.5819	0.5976	0.5741	27,339	24,803	30,963
Centroid Cosine Similarity	-0.1601	-0.0468	0.5760	0.6043	0.5699	29,576	22,441	32,304
FOAF ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	-0.1340	-0.0197	0.5750	0.5982	0.5698	29,972	24,513	32,355
Matrix Forest (α=0.001)	-0.1111	-0.0155	0.5839	0.5937	0.5769	26,842	26,143	30,402
Personalized SALSA (Auth;α=0.99)	-0.1379	-0.0489	0.5616	0.5849	0.5569	35,731	29,032	37,810
Local Path Index ($\Gamma_{out}(u)$;l=2;β=0.2)	-0.1479	-0.0360	0.5694	0.5872	0.5609	32,745	28,134	36,415
Pure Personalized PageRank (r=0.8)	-0.1408	-0.0344	0.5548	0.5706	0.5424	39,840	34,010	45,107
Jaccard ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.0123	0.0500	0.5861	0.5953	0.5837	25,884	25,786	28,039
Sorensen ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.0123	0.0500	0.5861	0.5953	0.5837	25,884	25,786	28,039
PropFlow ($\Gamma_{out}(u)$;l=2)	-0.1359	-0.0300	0.4498	0.4703	0.4501	81,422	66,784	45,293
TF-IDF ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.0121	0.0449	0.5780	0.5865	0.5754	29,580	28,841	31,716
Hannon ($\Gamma_{in}(u)$)	-0.0740	0.0020	0.5725	0.5871	0.5663	31,676	28,747	35,104

Table 56. Comparison of the different algorithms in terms of assortativity, modularity and weak ties (200 Tweets follows graph) (1 of 2)

Algorithm	Assortativity		Modularity			Weak ties		
	Undirected	Directed (IN)	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
Maximum Cosine Similarity	-0.1328	-0.0356	0.5573	0.5845	0.5516	37,998	29,298	40,603
Salton ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.0098	0.0479	0.5784	0.5901	0.5762	29,298	27,527	31,130
Hub Depressed Index ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.0140	0.0506	0.5852	0.5950	0.5819	26,206	25,903	28,791
Hub Promoted Index ($\Gamma_{in}(u);\Gamma_{und}(v)$)	-0.1260	-0.0547	0.5580	0.5739	0.5542	37,528	33,360	40,245
Money (Auth; k=50; α=0.3)	-0.1234	-0.0406	0.5094	0.5389	0.5080	57,136	43,788	55,617
Preferential Attachment ($\Gamma_{und}(v)$)	-0.1393	-0.0452	**0.4496**	**0.4696**	0.4498	81,544	67,030	78,078
Popularity	-0.1392	-0.0443	0.4572	0.4850	0.4518	82,125	62,774	78,179
PageRank (r=0.8)	-0.1379	-0.0434	0.4504	0.4713	**0.4489**	84,551	67,085	82,557
Personalized PageRank (r=0.8)	-0.1379	-0.0434	0.4504	0.4713	0.4489	84,552	67,085	82,557
LHN Index 1 ($\Gamma_{in}(u);\Gamma_{out}(v)$)	-0.0575	-0.0321	0.5406	0.5583	0.5305	45,497	38,827	51,877
Distance (Dir.)	-0.0806	-0.0175	0.5071	0.5403	0.4996	59,688	44,464	62,292
Katz ($\Gamma_{out}(u)$;β=0.4)	**-0.1733**	**-0.0650**	0.4579	0.4796	0.4496	84,076	66,040	87,845
LHN Index 2 (β=0.2)	-0.0404	-0.0154	0.5597	0.5835	0.5493	37,577	30,010	43,277
Hitting Time	-0.1331	-0.0544	0.4597	0.4775	0.4510	85,215	63,265	88,771
Commute Time	-0.1331	-0.0529	0.4597	0.4776	0.4509	85,209	63,249	88,927
Personalized HITS (Auth.;α=0.99)	-0.0650	0.0058	0.5586	0.5902	0.5610	3,711	2,418	2,588
Love(Auth.;k=1000;α=0.2)	-0.0651	0.0046	0.5586	0.5901	0.5609	3,679	2,426	2,584
PMF Basic (k=20.0;λ=0.01;ϕ=0.01)	-0.1126	-0.0500	0.5305	0.5503	0.5251	50,508	41,165	81,677
Random	-0.0054	0.0063	0.4581	0.4839	0.4525	80,964	64,057	83,292
PMF Sigmoid (k=20.0;λ=0.01;ϕ=0.01)	-0.1323	-0.0461	0.4565	0.4826	0.4539	81,695	64,498	83,707

Table 57. Comparison of the different algorithms in terms of assortativity, modularity and weak ties (200 Tweets follows graph) (2 of 2)

Algorithm	Comm-PairGini			Comm-InGini		
	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
ImplicitMF (k=300;α=40;λ=150)	0.5850	0.8404	0.0217	0.7765	0.9663	0.0921
User-based kNN (k=40)	0.5819	0.8364	0.0219	0.7637	0.9590	0.0917
Average Cosine Similarity	0.5769	0.8243	0.0213	0.7547	0.9690	0.0912
Adamic ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.5733	0.8251	0.0208	0.7469	0.9591	0.0895
QLJM ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;λ=0.1)	0.5811	0.8287	0.0216	0.7684	0.9698	0.0937
BM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.1;k=1)	0.5719	0.8249	0.0210	0.7515	0.9571	0.0903
BIR ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.5712	0.8223	0.0207	0.7460	0.9542	0.0889
Item-based kNN (k=300)	0.5804	0.8328	0.0219	0.7524	0.9654	0.0920
Res. Allocation ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$; $\Gamma_{und}(w)$)	0.5837	0.8371	0.0221	0.7755	0.9694	0.0980
ExtremeBM25 ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$;b=0.1)	0.5722	0.8269	0.0211	0.7550	0.9602	0.0909
Centroid Cosine Similarity	0.5689	0.8170	0.0215	0.7334	0.9562	0.0885
FOAF ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.5702	0.8204	0.0207	0.7373	0.9491	0.0878
Matrix Forest (α=0.001)	0.5921	0.8359	0.0235	0.7940	0.9602	0.0991
Personalized SALSA (Auth;α=0.99)	0.5603	0.8199	0.0209	0.7266	0.9376	0.0889
Local Path Index ($\Gamma_{out}(u)$;l=2;β=0.2)	0.5776	0.8209	0.0225	0.7597	0.9274	0.0931
Pure Personalized PageRank (r=0.8)	0.5875	0.8407	0.0228	0.8033	0.9540	0.0997
Jaccard ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.5917	0.8399	0.0237	0.7955	0.9699	0.1053
Sorensen ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.5917	0.8399	0.0237	0.7955	0.9699	0.1053
PropFlow ($\Gamma_{out}(u)$;l=2)	0.5105	0.7887	0.0240	0.5791	0.7781	0.1020
TF-IDF ($\Gamma_{und}(u)$; $\Gamma_{in}(v)$)	0.5941	0.8457	0.0243	0.8030	0.9604	0.1104
Hannon ($\Gamma_{in}(u)$)	0.5914	0.8293	0.0227	0.7948	0.9788	0.1023

Table 58. Comparison of the different algorithms in terms of community Gini (200 Tweets follows graph) (1 of 2)

Algorithm	Comm-PairGini			Comm-InGini		
	Louvain	Leading Vector	Infomap	Louvain	Leading Vector	Infomap
Maximum Cosine Similarity	0.5852	0.8402	0.0225	0.7667	0.9533	0.0996
Salton ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.5946	0.8418	0.0241	0.7983	0.9552	0.1063
Hub Depressed Index ($\Gamma_{und}(u);\Gamma_{in}(v)$)	0.5913	0.8398	0.0238	0.7920	0.9760	0.1052
Hub Promoted Index ($\Gamma_{in}(u);\Gamma_{und}(v)$)	0.5920	0.8415	0.0238	0.7687	**0.9816**	0.1043
Money (Auth; k=50; α=0.3)	0.5350	0.7788	0.0196	0.6645	0.8349	0.0788
Preferential Attachment ($\Gamma_{und}(v)$)	0.5110	0.7917	0.0165	0.5795	0.7802	0.0670
Popularity	0.5907	0.8434	0.0165	0.7095	0.8506	0.0670
PageRank (r=0.8)	0.5816	0.8427	0.0176	0.6910	0.8450	0.0742
Personalized PageRank (r=0.8)	0.5816	0.8427	0.0176	0.6910	0.8450	0.0742
LHN Index 1 ($\Gamma_{in}(u);\Gamma_{out}(v)$)	0.6050	0.8554	0.0303	0.7921	0.9643	**0.1408**
Distance (Dir.)	0.5986	0.8551	0.0229	0.7829	0.9588	0.0957
Katz ($\Gamma_{out}(u);\beta$=0.4)	0.5838	0.8873	0.0222	0.7014	0.9407	0.0935
LHN Index 2 (β=0.2)	0.5952	0.8546	0.0283	0.8021	0.9570	0.1262
Hitting Time	0.5006	0.6902	0.0169	0.5932	0.6700	0.0715
Commute Time	0.5005	0.6903	0.0181	0.5931	0.6702	0.0769
Personalized HITS (Auth.;α=0.99)	0.5678	0.8517	0.0184	0.7747	0.9678	0.0930
Love(Auth.;k=1000;α=0.2)	0.5679	0.8517	0.0184	0.7749	0.9677	0.0930
PMF Basic (k=20.0;λ=0.01;ϕ=0.01)	0.6100	0.8575	0.0261	**0.8228**	0.9499	0.1010
Random	0.6355	**0.8999**	**0.0321**	0.7913	0.9663	0.1217
PMF Sigmoid (k=20.0;λ=0.01;ϕ=0.01)	**0.6395**	0.8910	0.0223	0.7952	0.9591	0.0945

Table 59. Comparison of the different algorithms in terms of community Gini (200 Tweets follows graph) (2 of 2)